Dvé: Improving DRAM Reliability and Performance On-Demand via Coherent Replication

Adarsh Patil
University of Edinburgh
adarsh.patil@ed.ac.uk

Vijay Nagarajan
University of Edinburgh
vijay.nagarajan@ed.ac.uk

Rajeev Balasubramonian
University of Utah
rajeev@cs.utah.edu

Nicolai Oswald
University of Edinburgh
nicolai.oswald@ed.ac.uk

***Abstract*—As technologies continue to shrink, memory system failure rates have increased, demanding support for stronger forms of reliability. In this work, we take inspiration from the two-tier approach that decouples correction from detection and explore a novel extrapolation. We propose Dvé, a hardware-driven replication mechanism where data blocks are replicated in 2 different sockets across a cache-coherent NUMA system. Each data block is also accompanied by a code with strong error detection capabilities so that when an error is detected, correction is performed using the replica. Such an organization has the advantage of offering two independent points of access to data which enables: (a) strong error correction that can recover from a range of faults affecting any of the components in the memory, upto and including the memory controller, and (b) higher performance by providing another nearer point of memory access. Dvé realizes both of these benefits via Coherent Replication, a technique that builds on top of existing cache coherence protocols for not only keeping the replicas in sync for reliability, but also to provide coherent access to the replicas during fault-free operation for performance. Dvé can flexibly provide these benefits on-demand by simply using the provisioned memory capacity which, as reported in recent studies, is often underutilized in today's systems. Thus, Dvé introduces a unique design point that offers higher reliability and performance for workloads that do not require the entire memory capacity.**

***Index Terms*—memory systems, DRAM, reliability, coherence**

I. Introduction

For servers or systems with high-value data like in finance, automotive, and healthcare, system reliability is of utmost importance. Improving memory reliability has been key to improving overall system reliability. Field studies of memory in datacenters [45], [64], [65] and supercomputers [4], [24], [67], [68] have reported patterns of larger granularity memory errors/failures and unexpected DRAM failure modes [38], corroborating the need for improved memory reliability.

There have been several techniques and proposals for improving the fault tolerance of DRAM memory. These works have largely focused on incrementally increasing the efficiency and scope of error control mechanisms in memory subsystems. DRAM memory schemes initially only targeted cell failures and progressively evolved to handle chip, DIMM, and channel failures. Primarily, these schemes pad data with error correcting code (ECC) and distribute the resulting codeword on a set of components such that, on a partial failure, data can be recovered from the remaining fault-free data and the padded error correction code.

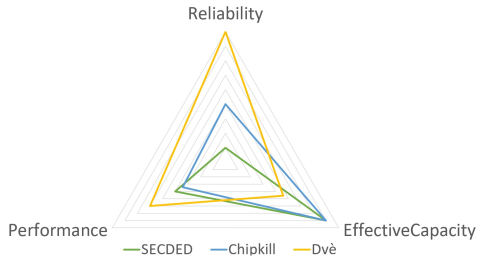

Fig. 1. Comparison of various DRAM reliability designs

One notable step in this progression is the recent body of work that has advocated the decoupling of error detection from correction by breaking down DRAM fault tolerance into two tiers [31], [33], [43], [50], [73], [79]. These works add a check code per codeword for detecting errors in the first stage and an error correction mechanism at a larger granularity in the second stage. Doing so allows for deploying more powerful ECC codes to recover from a larger class of errors. This decoupling also allows storing ECC bits elsewhere in memory and thus does not impose restrictions on the configuration and operation of DRAM DIMMs.

In this work, we take inspiration from the two-tier approach and explore a novel extrapolation. We propose Dvé[1], a hardware-driven replication mechanism with the following important features.

1) Dvé leverages ECC codewords and other existing mechanisms for error detection but provides recovery from a detected error by reading from a replica of the data, instead of reconstructing data using the ECC bits.
2) It significantly improves memory reliability by keeping replicas as far apart and disjoint as possible – replicating data across 2 different sockets on the same system, thereby tolerating errors anywhere in the entire memory path (controllers, channels, DIMMs, and DRAM chips).
3) Dvé introduces *Coherent Replication*, a technique that builds on top of existing cache coherence protocols for not only keeping the data and replica in sync, but also providing coherent access to the replicas during fault-free operation. In doing so, Dvé alleviates some of the NUMA latency overheads as data can be accessed at the nearest replica memory. It also provides improved

[1]Dvé (Sanskrit) translates to *the two*, referring here to the dual benefits of replication.

memory access bandwidth by providing two endpoints to access data.

4) Dvé can be employed on-demand by taking advantage of the memory that is often underutilized in large installations ([28], [42], [56], [58]), thus allowing flexibility between capacity and reliability.

A New Tradeoff. Fig. 1 compares DRAM RAS mechanisms: SEC-DED (bit level error protection), Chipkill (chip level error protection), and Dvé across the goodness metrics of reliability, performance and effective capacity (inverse of capacity overheads).

Dvé achieves higher reliability (at least 4× lower uncorrected error rate than Chipkill) as its design is more robust to failures and can detect/correct a larger granularity of errors. The only Achilles heel for Dvé is in the case of simultaneous failure in exactly the same location on a pair of completely independent replicated memory components, the occurrence of which is lower in probability than 2 DRAM devices failing in the same memory rank as in the case of Chipkill. Thus, Dvé provides stronger protection against memory errors.

Typically, error detection/correction imposes a performance overhead. Many manufacturers concede that Chipkill ECC DRAM will be roughly 2-3% slower than non-ECC DRAM [62]. Although Dvé still requires error detection, by providing two independent points of access to the data, it provides performance improvement in a multi-socket NUMA organization.

Dvé uses simple data replication with higher capacity overheads (lowering effective capacity to 43.75% compared to 85% for Chipkill). While the capacity overheads for Chipkill are strictly fixed at design time, Dvé overheads are applicable only when employed on-demand at runtime (for example, when memory is underutilized).

Contributions. We introduce Dvé, a hardware-driven replication mechanism which provides the dual benefit of improved reliability and performance. Specifically:

- We explore a unique design point which trades off reduced memory capacity for higher reliability and performance by replicating data blocks across two sockets of a cache-coherent NUMA system
- We analytically quantify Dvé's reliability benefits and show that it provides lower uncorrectable and undetectable error rate over Chipkill ECC and thus provides higher memory reliability. Similarly, Dvé in conjunction with Chipkill ECC provides 2 orders of magnitude higher reliability over IBM RAIM [44]. Further, Dvé's thermal risk aware mapping lowers DUE by at least 11% over Intel memory mirroring [26].
- We propose Coherent Replication, a technique that builds on top of existing protocols to not only maintain the replicas in sync (required for reliability), but also provide coherent access to both of the replicas during common-case fault-free operation (for performance).
- To allow for flexibility between capacity and reliability, we propose a hardware-software co-design approach to enable/disable replication when desired at runtime.

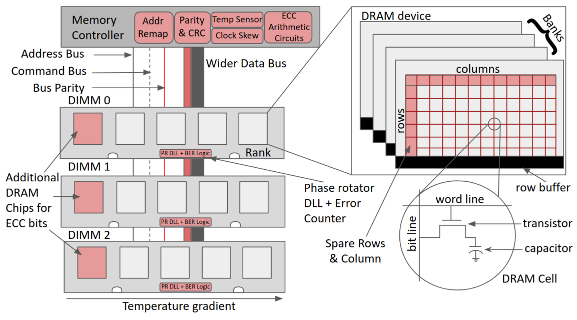

Fig. 2. Anatomy of RAS features in memory

- Our experiments indicate that Dvé provides performance improvements of between 5%-117% across 20 workloads over a dual-socket NUMA system, and between 3%-107% over an improved (hypothetical) version of Intel's memory mirroring scheme.

II. Motivation

We first motivate the need for improved DRAM reliability to combat: (a) projected increase in errors caused by DRAM design trends; and (b) DRAM failures caused by *any* part of the DRAM subsystem. Secondly, we motivate the need (and opportunity) for providing improved memory reliability *on demand*. Finally, we summarize by identifying the limitations of existing approaches.

A. Growing DRAM Error Rates

The memory subsystem has several modules built-in at various points in the hierarchy (e.g., cell, rank, bank, chip, memory controller) to improve reliability of the DRAM as highlighted in red in Fig. 2. We analyze these current DRAM specs/chips and state-of-the-art error protection mechanisms and observe that these would be inadequate for handling the nature of faults/errors seen in field studies.

Cell errors and their increasingly costly mitigation. DDR5 DRAM chips are expected to have 4 times the memory capacity per DRAM chip, in line with the trend of miniaturization and higher density of DRAM. To combat the increase in cell fault rates due to smaller cell geometry, increased variability of manufacturing, and additional refresh pressure, DDR5 includes simple in-DRAM (on-die) ECC [47]. DDR5 DIMMs have also doubled the number of error correcting bits compared to DDR4 DIMMs, i.e., from 8-bit to 16-bit ECC for 64-bit data (25%) [47]. Several mechanisms already propose row/column sparing and selective replication [10], [52] to tolerate higher number of faulty cells caused during manufacturing. The capacity overheads (provisioned invisible redundant capacity) required to maintain reliability is growing.

Non-cell errors are becoming important. Studies have shown that unpredictable large multi-bit DRAM failures can occur due to faults in the chip-internal circuitry [67] that can affect multiple banks, rows, etc. within a DRAM chip. To handle such DRAM chip errors, several variants of Chipkill [2] solutions have been developed. Further studies have observed shared board-level circuitry failures that cause cascading errors, rendering multiple DRAM chips (that share circuitry within a DIMM) erroneous or inaccessible [30]. To cope with such failures Bamboo ECC [36] and Virtualized ECC [79] were proposed to handle multi-pin and multi-chip failures. In addition, studies have shown that faults outside the DIMM such as faults in memory controller logic that interacts with the external DRAM subsystem, errors in the channel, or electrical disturbances also affect the reliability of DRAM memory [37], [45], [65]. These studies suggest that a wide variety of memory failures can occur and existing mechanisms are insufficient to handle these errors because they co-locate data and correction codes on the same channel.

To reduce channel errors, DDR5 memory uses a host of bus reliability mechanisms like command/address parity checks, bus CRC, gear down mode. DDR5 chips are to feature delay-locked loop and forward feedback equalization to handle channel errors that occur because of the higher DDR frequencies [68]. These bus error checks only detect errors and perform transaction retry; they cannot tolerate hard channel errors.

Stronger ECC codes with longer codewords were introduced to detect and correct channel errors in [31], [37]. Increasing the codeword length is also problematic, as the decoder complexity increases more than linearly with the codeword length [7]. Further, sophisticated techniques use 2 channels in a RAID-1 layout (Intel Memory Mirroring [26]) or 5 "ganged" channels in RAID-3 layout (IBM RAIM [44]) to tolerate complete channel failures. However, each of these techniques limits reliability and performance benefits. RAIM's ganged channel mode forces 256 byte memory reads and writes which negatively impacts performance [82] and leaves it susceptible to any errors in the single RAIM controller. Although Intel's mirroring scheme replicates memory between channels within a controller, the secondary channel's copy is used as a backup and is read only on the failure of the primary – thus, providing no performance benefits despite the existence of data replicas. Further, Intel's approach localizes replicas to a single socket and a single memory controller leaving it susceptible to any faults in the controller or its subsystem. Additionally, the memory that is replicated is fixed at boot time and limited to the OS kernel memory.

DRAM reliability is also impacted by external factors like temperature, requiring sufficient timing slack margins while operating DRAMs [40] or throttling of requests to avoid thermal emergencies [41]. DRAM disturbance faults or row-hammer faults [38] demonstrate that new and unexpected multi-bit failures may occur while in operation. Mitigations include more frequent memory refresh for frequently accessed rows which could cause performance degradation.

[2]Chipkill is a generic term for a solution that guarantees recovery from failure of an entire DRAM chip.

B. Need for on-demand memory reliability

The mitigation techniques thus far have fixed area and logic overheads (at design time) for providing memory reliability. We observe the possibility of using idle memory capacity to opportunistically improve memory reliability. To accomplish this, we motivate the need for such a reliability service to be flexible and allocatable on-demand.

Large scale memory underutilization is prevalent. Several studies point to the phenomenon of memory underutilization in HPC systems [28], [56] and in cloud datacenters (e.g., Alibaba- [14], [42], Google- [58]). These works report that at least 50% of the memory is idle 90% of the time. There exists a large gap between a node's maximum/worst-case memory utilization and the common-case memory utilization; i.e., a few workloads have high utilization while most other workloads have significantly lower utilization. However, memory resources are often over-provisioned due to peak estimation. Datacenter operators procure systems with a view to keeping the machines homogeneous with respect to the workloads or to improve the system's capability to solve large problem sizes, which is an important figure of merit in HPC systems [56].

The bulk of applications, that are not memory capacity intensive, would benefit from increased memory reliability and improved memory access latency [56]; for example, capacity-agnostic long running HPC applications. Furthermore, some applications may require reliability for only a small region of memory; for example, cloud applications need fault-tolerance for only the stateful memory regions and not the stateless regions. Providing this flexibility between capacity and reliability allows deploying large numbers of commodity DRAMs or lower reliability DRAMs for high capacity while still being able to turn on/off higher reliability on demand.

A central observation in our work is that servers underutilize their memory capacity, and applications can exploit this idle memory to boost performance and reliability. Industry products like Intel Memory Mirroring [26], that relinquish half their memory capacity for high reliability, confirm that this observation is valid, but has only been partially exploited.

Error Rates increase as DRAMs age. Another need for on-demand reliability is to combat the higher error rates observed as DRAMs age and suffer from wear-out faults [18]. This is due to degradation of retention time and increased sensing delays. Memory systems today do not allow for flexibly boosting reliability, requiring periodic memory replacement.

Summary. State-of-the-art DRAM error protection mechanisms suffer from the following limitations.

1) Existing mechanisms jeopardize the correction capability by putting correction mechanisms in the same "basket" as data. Given that failures can occur at any level in the memory subsystem, current mechanisms are therefore vulnerable to failures beyond a channel, including errors in the memory controller.
2) They trade off reduced error detection capability for some amount of correction capability which limits their effectiveness to detect more errors (e.g., designing for

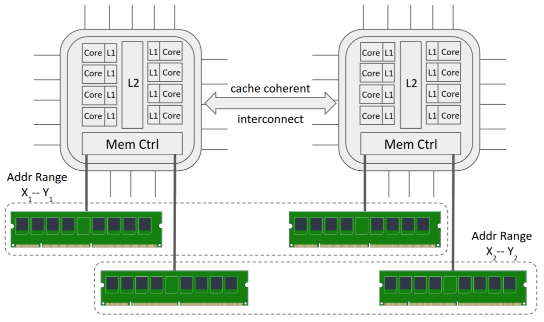

Fig. 3. Dvé Replication Schematic

Double Symbol Correction before Triple Symbol Detection).

3) Existing mechanisms typically impact performance negatively. Using Chipkill ECC DRAM reduces performance by 2-3% [62] over non-ECC DRAMs. Even contemporary reliability techniques like Virtualized ECC [79] and Bamboo ECC [36] reduce performance further by 3-9% and 2-10% respectively. Further, alleviating temperature induced effects and row hammer mitigations tend to hurt performance.
4) They lack flexibility to provide memory reliability on demand by adapting to workloads requirements.

III. Dvé: Overview

The variety and the granularity of DRAM errors are increasing. Correcting all of these errors demands a robust mechanism. We argue for a broadsword approach to error correction that is decoupled from error detection, and can correct errors of *any* granularity. We advocate for changing the perspective of DRAM protection from an incremental, short-sighted view to a holistic approach leveraging the time-tested end-to-end argument. We argue for protecting memory at the highest end point of memory (i.e., at the memory controller level), thereby subsuming all other types of errors.

Our solution, Dvé is a hardware-driven replication scheme for achieving not only reliability but also performance. Dvé performs memory replication on two memory controllers located on different cache-coherent NUMA nodes (as shown in Fig. 3); when a dirty block is written back to its home memory node, it is also written to its replica. Using a different "basket" for recovery allows us to recover from a wide variety and granularity of failures. Indeed, Dvé can tolerate large multi-bit errors due to memory controller logic failure, bus failures as well as any failures in shared components in the hierarchy.

Because Dvé relies on a replica for correction it needs to store only error detection codes. Therefore, it requires only error detection circuitry that is simpler to build as it involves computation of just the error-locator polynomial (error correction also involves computing the extra error-evaluator polynomial for symbol based codes [6]). The extra code space available as a result of forgoing the correction code can be used to store stronger detection codes for detecting larger number and/or larger granularity of errors. Along with ECC based detection, Dvé can rely on any new and/or existing fault detection techniques, such as CRC or parity present in the DDR4 spec [60], and additional hardware and firmware diagnostic capabilities like temperature sensors, clock skew detection, etc., to mark failed components (Fig. 2).

Dvé's replication proves advantageous in several other scenarios as well. Mapping replicated data onto DIMMs with different thermal properties – e.g., data on a hot DIMM/chip replicated on a relatively cooler DIMM/chip on the other socket – ensures reduced temperature induced failures. Row hammer errors can be mitigated by load balancing requests between the independent replicas. Sec. IV quantifies the reliability benefits of Dvé using failure rates from field studies.

While performance penalties are a problem for existing schemes, in Dvé, the presence of the replica in another NUMA node provides an opportunity to boost performance. Note that in order to ensure strict recovery semantics, the data and its replica needs to be kept consistent at all times. Happily, a replica that is kept strongly consistent allows for the replica to be accessed by reads even during fault-free operation. In other words, it allows for a read request to be potentially serviced from the nearest replica to mitigate some of the NUMA latency overheads and also improve memory bandwidth. In Dvé, we realize this via Coherent Replication (Sec. V-C), a technique that extends existing coherence protocols to keep the data and the replica consistent, while providing coherent access to both the data and the replica during fault-free operation.

In Dvé, each physical address is mapped to a replica physical address. This can either be a fixed function mapping or a flexible table-based mapping. A flexible mapping allows for providing memory reliability on demand and requires the OS to map each allocated physical page to a replica physical page. Using the OS memory allocator's understanding of the system's memory topology, replica page pairs are made such that they exist on memory adjoining different sockets. A single system-wide OS managed *replica map table* (RMT) maps a physical page to its corresponding replica page. If an entry does not exist in the RMT, Dvé seamlessly falls back to using a single copy. On the other hand, a fixed function mapping[3] benefits from fast translation to replica address and works well if the entire memory space is being replicated en masse.

Unless stated otherwise, we assume that all memory is replicated using a fixed function mapping and that there is one replica for every data item (i.e., 2 copies). The core workings of Dvé are unchanged even with a table based mapping, which would require an additional lookup into the RMT to locate the replica address. We discuss the details of such a flexible mapping system in Sec. V-D.

[3]a fixed mapping is a static direct-mapped function of the form f : $p(S, Ro, Ra, Ba, Co, Ch) \rightarrow p_r(S', Ro', Ra', Ba', Co', Ch')$, $\forall p, p_r \in P$ where p, p_r are physical addresses in P and S, Ro, Ra, Ba, Co, Ch correspond to socket number, row, rank, bank, column, channel respectively. An example for such a function which we use in this work, is given by $f(p) = \frac{p}{L} + 1 - (2 * S)$ where L is page size. The function considers consecutive physical pages interleaved between sockets and maps to a replica page on the other socket but retains the same DRAM internal mapping.

TABLE I
DUE AND SDC RATES (PER BILLION HOURS OF OPERATION) AND IMPROVEMENT †TEMPERATURE SCALED FIT RATE

Scheme	DUE		SDC	
	Rate (lower is better)	Impr.	Rate (lower is better)	Impr.
Chipkill	10^{-2}	–	3.1×10^{-10}	–
Dvé+DSD	2.5×10^{-3}	4×	6.3×10^{-10}	0.49×
Dvé+TSD	2.5×10^{-3}	4×	2.5×10^{-16}	~10^6×
IBM RAIM	1.5×10^{-14}	–	4.0×10^{-10}	–
Dvé+Chipkill	8.7×10^{-17}	172×	6.3×10^{-10}	0.63×
Chipkill†	2.2×10^{-2}	–	1.0×10^{-9}	–
Intel+TSD†	5.9×10^{-3}	3.72×	1.1×10^{-15}	~10^6×
Dvé+TSD†	5.3×10^{-3}	4.15×	1.1×10^{-15}	~10^6×

IV. Quantifying the Reliability of Dvé

Dvé's robust design can recover from a large breadth of memory related errors that can be detected. This is because Dvé can simply adopt differing bits from the replica, recompute the code, and confirm it matches. This provides asymptotically better reliability than any ECC based correction scheme. We now quantify the reliability improvements.

DRAM reliability mechanisms use Forward Error Correction (like ECC) which add redundant information so that data can be recovered when errors are encountered. Block codes that work on fixed-size blocks or "symbols" are used to allow encoding/decoding in polynomial time. Various classical block codes such as Hamming codes, Reed-Solomon codes, BCH codes apply the algebraic properties of Finite (Galois) Field Arithmetic to correct and detect errors in DRAM. These error control systems can have one of the following outcomes: (a) corrected error (CE), (b) detected but uncorrected (DUE) error, or (c) suffer Silent Data Corruptions (SDC).

Comparative Case Studies: For this, we analytically model (similar to [70]) and quantify reliability improvements of Dvé using DUE, SDC rates with a uniform DRAM device FIT rate of 66.1 [67]. Since Dvé can use *any* detection code, we equip Dvé with a similar detection capability as the scheme being compared against.

A. Comparison to Chipkill ECC

Consider a system with 32 single rank ECC DIMMs, each DIMM containing 9 DRAM chips. The baseline Chipkill ECC can tolerate one failed chip per rank[4]. For Dvé the 9th chip is modeled in 2 ways:
(i) equipped with detection code similar to the baseline (DSD)
(ii) equipped with stronger detection code (TSD)[5]; using the extra capacity obtained by relinquishing the correction code present in the baseline
DUE rate: A Chipkill system fails to correct an error if 2 chips fail simultaneously within a single DIMM[6] inside a scrub interval which is given by $[9 \times 66.1 \times 8 \times 66.1 \times 10^{-9}] \times 32$ ($\approx 10^{-2}$) in every billion hours of operation.

Each model imposes certain constraints on the chip failures that are uncorrectable; more constraints lead to a lower uncorrectable rate. In Dvé, the system fails to correct if 2 corresponding chips *in the same position* on two DIMMs in the corresponding rank fail together inside a scrub interval which is given by $[9 \times 66.1 \times 1 \times 66.1 \times 10^{-9}] \times 32 \times 2$ ($\approx 2.5 \times 10^{-3}$) per billion hours of operation.

Thus, Dvé provides $4\times$ lower DUE rate than a Chipkill system. It is worth noting that this number is irrespective of the detection code and is only a factor of the number of replicas.
SDC rate: A Chipkill system potentially fails to detect an error if three or more chips fail simultaneously within a DIMM inside a scrub interval. A simultaneous three device failure probability is given by $[9 \times 66.1 \times 8 \times 66.1 \times 10^{-9} \times 7 \times 66.1 \times 10^{-9}] \times 32$ ($\approx 4.6 \times 10^{-9}$). The probability of DSD code failing to detect three chip failure is 6.9% [77]. Thus, overall SDC is alteast $(4.6 \times 10^{-9} \times 0.069)$ per billion hours of operation.

For Dvé the SDC rate using a DSD code is twice that of Chipkill since we use double the number of DIMMs for replication and a SDC error can occur in either replica. However, with a TSD code this number reduces drastically as the detection potentially fails only if four or more chips fail simultaneously within a single DIMM.

For even better detection options, low-overhead highly-efficient codes [6] which come closer to reaching the theoretical Shannon limit can be employed. Alternatively, incremental multi-set log hashes [13] can also be used to detect errors. We leave such options for future work.

B. Comparison to IBM RAIM

While Chipkill ECC was not designed to tolerate channel failures, a high reliability system such as IBM RAIM provides a more outright design point for comparison. Recall RAIM uses Chipkill ECC DIMMs with RAID-3 organization across 5 channels; striping four cache lines across four channels and adding redundant diff-MDS ECC code [39] in the fifth channel to correct upto 1 entire channel failure.

We assume 5 RAIM channels each with 8 Chipkill ECC DIMMs and Dvé equipped with 2 replicated channels with 32 Chipkill ECC DIMMs each on different NUMA nodes.
DUE rate: RAIM fails to correct an error if 2 two corresponding Chipkill DIMMs on 2 channels (out of the 5 channels) fail together. Thus, the DUE is calculated as $[(1^{st}\ Chipkill\ DUE \times 8) \times (4) \times (2^{nd}\ Chipkill\ DUE \times 1)] \times 5(\approx 1.5 \times 10^{-14})$ per billion hours of operation.

Dvé+Chipkill fails to correct an error only if 2 pairs of chips in the same position on two DIMMs fail together which is given by $[9 \times 66.1 \times 8 \times 66.1 \times 10^{-9} \times 1 \times 66.1 \times 10^{-9} \times 1 \times 66.1 \times 10^{-9}] \times 32 \times 2$ ($\approx 8.79 \times 10^{-17}$) per billion hours of operation. Hence, Dvé+Chipkill provides $172.4\times$ *(2 orders of magnitude)* lower DUE than RAIM.
SDC rate: Both systems suffer a SDC when Chipkill ECC fails to detect an error. In addition, RAIM also potentially suffers an

[4]Assuming 8-bit symbol based RS(18,16,8) code with SSC-DSD (Single Symbol Correct-Double Symbol Detect), organized as in Virtualized ECC [79]
[5]Triple Symbol Detect (TSD) is provided using 16-bit Reed-Solomon code as in Multi-ECC [31]
[6]Chipkill ECC is per rank. This being a single rank DIMM, failure of the rank implies failure of the DIMM.

SDC when 3 channels fail simultaneously. (Probability of this is significantly lower and hence both are limited by Chipkill ECC SDC). Since the total number of DIMMs in Dvé is higher Dvé+Chipkill (64 DIMMs, compared to 40 in RAIM) it has a marginally higher SDC compared to RAIM.

C. *Thermal effects on Reliability*

To factor in temperature effects on reliability we scale the FIT rate using Arrhenius Equation [48]. There exists a 10°C temperature gradient between the DRAM chip closest and farthest to the fan [41], leading to non-uniform FIT rates for the 9 chips within a DIMM scaled as [66.1, 74.3, 82.5, 90.7, 98.9, 107.1, 115.3, 123.5, 131.7]. Using a similar calculation as above, we see that although overall DUE and SDC increases for baseline Chipkill system, Dvé+TSD is able to lower DUE by $4.15\times$ and provide significant reduction in SDC compared to the temperature-factored Chipkill baseline by using a risk inverse mapping (data in chips that have higher FIT rate are mapped to chips in the replica that have lower FIT rate).

When compared to an Intel mirroring-like scheme (employing TSD for a fair comparison), Dvé+TSD is able to lower DUE by 11% using the thermal risk inverse mapping while the Intel mirroring scheme, despite the presence of replicas, does not. While the analysis above exploits a non-uniform thermal distribution across chips in a rank, some boards may exhibit a non-uniform thermal profile across ranks, e.g., ranks closer to the processor may exhibit higher temperatures than ranks further from the processor. Memory controller policies can be designed to place the two copies of data in ranks that are not both at high risk of failure, thus achieving higher overall reliability than thermal-unaware policies – we leave such explorations for future work.

D. *Summary*

Table I summarizes the DUE & SDC rates. A key reason why Dvé provides significantly higher reliability over Chipkill and RAIM is because Dvé relies on full replication, while other schemes are all based on ECC (which is a “k-out-of-n” system). More precisely, our competitors rely on $(n-k)$ out of n hardware entities operating correctly – where hardware entities can be chips, channels etc., and $k = 1$ or 2 and n is between 5 and 9, typically. In contrast, Dvé only relies on the exact same hardware entity in the other independent replica not failing. Therefore there are more ways for our competitors to fail compared to us. Finally, it is worth noting that our analytical model does not account for other memory subsystem failures like those in address/data bus, memory controllers, etc., due to absence of field data for these. Because Dvé is the only scheme that can tolerate such failures, our analyses serve as lower bound for the actual DUE, SDC rates.

V. Dvé

In this section, we describe how Dvé achieves both reliability and performance via replication. We first outline the system model after which we precisely specify the consistency and recovery semantics of Dvé. Then, we describe details of the coherence protocol extensions for realizing these semantics. Finally, we discuss how a flexible replica region is organized.

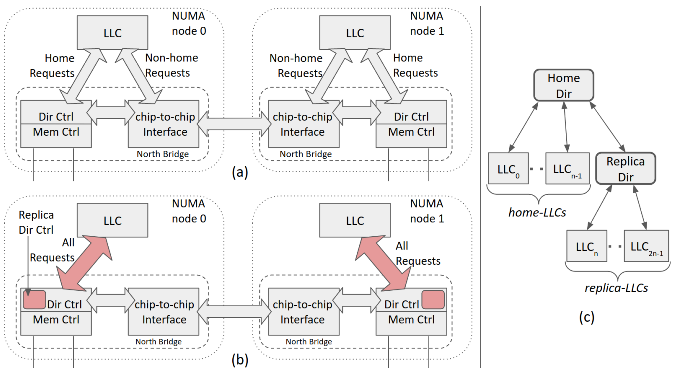

Fig. 4. Coherence in (a)NUMA(above) (b)Dvé(below) (c)Logical view(right)

A. *System Model*

We assume a typical modern system consisting of multiple multi-core chips connected via a cache-coherent, high-bandwidth, low-latency point-to-point interconnect like Intel QPI, UPI or AMD Hyper-transport. Each multi-core chip seated within a socket has a DRAM memory array co-located with it. Each chip has multiple levels of SRAM caches including a last-level cache (LLC) that is shared by the cores within that socket, but globally the LLC is private to each socket as shown in Fig. 4(a). On an LLC miss, the request is routed to the “home directory” adjoining the physical location of the home memory controller. (The home directory for an address is determined based on a static hash function of the address.) Thus, the memory access latency depends on where the request originated and where the memory is located. Accessing locations mapped to a remote socket experiences a higher latency, as they require traversing one or more socket interconnect links compared to accessing locations on the same socket. A hierarchical cache coherence protocol handles permissions for each write request and enforces write serialization. Coherence is enforced by looking up the logically centralized (but physically distributed) global directory. We assume a full directory with the recently accessed entries cached on-chip [66].

Dvé uses either a statically reserved portion of the entire memory space for replication with a fixed function mapping or employs a flexible table based mapping populated by the OS as explained in Sec. III. For either case, blocks are always inserted into the caches using the original physical address and only the directory controller is responsible for maintaining consistency between the replicas. Note that there are no requests from caches for addresses in the replica pages since these are unused/unallocated by the OS.

B. *Consistency and Recovery Semantics*

1) Consistency: Dvé extends the coherence protocol to: (a) keep the replica strongly consistent with the data; and (b) ensure that the replica is accessible during fault-free operation.

To maintain a strongly consistent replica, when a dirty cache block is replaced from the LLC, the block is written back to the home node as well as the replica, synchronously. (By "synchronously", we mean that the request completes only after both home node and replica are written to.) A strongly consistent replica is a necessary but not sufficient condition to ensure that the replica can be accessed during fault-free operation. The data in memory, and hence the replica, could be stale when some cache in the system holds the location in writable state. Therefore, we augment the coherence protocol with additional metadata and logic (in the form of a *replica directory*) to ensure that the replica is accessed only when it is not stale. With these extensions, Dvé ensures that a read request can be serviced from nearest replica.

2) Recovery: Dvé's strong consistency guarantee makes recovery straightforward. When a memory read fails in one of the replicas, i.e., the local ECC check (if equipped with DSD/TSD) or local ECC check+repair (if equipped with Chipkill) at the memory controller fails after a DRAM read, the home/replica directory diverts the request to the other memory controller for recovery. (If the other copy's read also fails, the data is lost (DUE) and a machine check exception is logged and signaled.) If the copy is good, data is returned to the requester and the system logs a Corrected Error (CE). The initial memory controller attempts to fix its copy by updating (writing) it with the correct data and then re-reading the DRAM. If the error was temporary, this read will succeed else the system is placed in a *degraded state* with only one working copy.

C. Coherent Replication

Logically speaking, Dvé introduces a new replica directory and so, each location now has a home directory as well as a replica directory. In physical terms, Dvé augments each directory controller with metadata (as shown in Fig. 4(b)) to allow for the replica values held in that socket to be safely accessed. Normally each physical directory maintains state for only locations mapped to that node. In Dvé each directory also maintains state about replica locations mapped to that node.

For each location, all LLCs in the system can be classified into two classes: (a) *home-LLCs*: LLCs that send their request to the home directory – the home directory being nearer to them; and (b) *replica-LLCs*: LLCs that send their request to the replica directory – the replica directory being nearer. (Note that LLCs can still cache any block in system memory.) There is a hierarchical relationship between home directory and replica directory as shown in Fig. 4(c). The replica-LLCs view the replica directory as a cache of the home directory. Transactions originating from the replica-LLCs check the replica directory first before going to the home directory. The home directory on the other hand views the replica directory as one of its "sharers"; it forwards requests to the replica directory which in turn consults its own sharer vector and forwards the requests to one or more of the replica-LLCs.

We propose two protocol families – allow-based and deny-based – based on how access permissions are acquired for

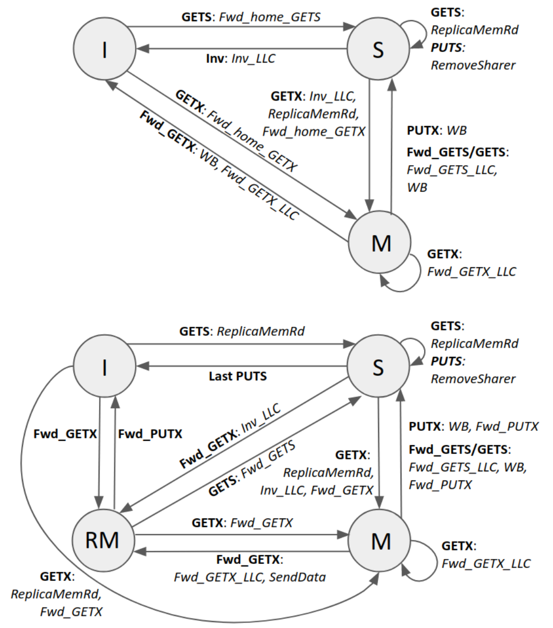

Fig. 5. Replica directory controller: stable states and transitions. *Above:* allow-based protocol; *Below:* deny-based protocol

replicas. In the former, the replica directory pulls "allow permissions": replica can be accessed only if a replica directory entry for that location explicitly says the location can be accessed; the absence of an entry means "no". In the latter, the home directory pushes "deny permissions" to the replica directory: replica can be accessed unless a directory entry explicitly forbids its access; absence of an entry means "yes".

1) Allow-based Protocol: The replica directory maintains entries like in a conventional director – including state, sharer vector, and owner. Without loss of generality we assume the MSI states. "Invalid" means that the location is not cached in any of the replica-LLCs. "Shared" means that the location is cached in readable state in one or more of the replica-LLCs. "Modified" means that the location is cached in writable state in one of the replica-LLCs.

Suppose that there is an LLC read miss in a socket that is sent to the replica directory – the socket being closer to the replica directory. The request can safely read from the replica if (and only if) that location is in shared state in the replica directory. If it is in modified state, it has to be routed to the owner in one of the replica-LLCs.

Importantly, if the entry for the location does not exist (i.e., the location is in invalid state) the replica cannot be safely accessed because it is possible that one of the home-LLCs may currently hold the block in modified state. In such a case, the request is forwarded to the home directory. The home directory responds to the request with the value and adds the replica directory as one of its "sharers". Upon receiving the response, the replica directory goes into shared state and sets the sharer vector to point to the LLC that initiated the request.

The complete state transition diagram of the replica controller is illustrated in Fig. 5 (top). The states and transitions resemble a conventional MSI directory controller but with one crucial difference. When one of replica-LLCs evict a dirty block in modified state, the replica directory not only writes back the block to the replica memory but also the home memory. In a similar vein, the home directory also writes back a dirty block to both the home memory and replica memory.

In summary, the allow-based family of protocols lazily pull read permissions for the replica upon access. This reactive approach works well on workloads with significant private writes; it makes sense to avoid pushing permissions to the replica directory via the inter-socket link when other threads are not likely to read those lines.

2) Deny-based Protocol: In contrast to allow-based, in the deny-based family of protocols, the home directory eagerly pushes deny permissions (i.e., knowledge about writable locations in their LLCs) to the replica directory. In doing so, the replica memory can be accessed even if there is no directory entry corresponding to that location. The proactive approach is suited to read-only (or mostly-read) workloads since directories can read the nearest replica without the need for requesting permissions.

Like in the allow-based protocol, the replica directory again maintains entries like in a conventional protocol including: state, sharer vector, and owner. States include the conventional MSI states, but additionally a new *remote modified* (RM) state. A location in RM implies that one of the home LLCs have the block in modified state; this implies that the replica is stale and hence cannot be accessed directly. "Invalid","Modified" and "Shared" states mean the same as in the allow-based protocol.

Suppose that there is an LLC miss that is sent to the replica directory. The request can be safely read from the replica as long as the location is not in RM state. Note that the replica can be read even if there is no entry in the replica directory. Indeed, the absence of an entry implies that there are no remote writers and hence means that the replica is not stale.

Finally, as in the allow-based protocol, an evicted dirty LLC block is written to both the home memory and the replica memory. The state diagram of the replica directory controller is illustrated in Fig. 5 (bottom).

3) Handling Recovery: The recovery process of Dve is simple and does not involve any stop-the-world state update: when a local memory controller returns a read failure, the directory simply forwards the request to the replica memory controller. In other words, the coherence protocol is agnostic to the recovery action. Any concurrent request from a cache or an I/O operation is serialized and coalesced at the directory in the MSHR/intermediate state as in the baseline coherence protocol. This invariant ensures correctness for all cases.

4) Complete protocol and Verification: Until now we have discussed only stable states and transitions, implicitly assuming that state transitions happen atomically. In reality, each transition involves a number of steps in modern systems. For this reason, transient states and actions are necessary to enforce logical atomicity. We have fully fleshed out complete protocol specifications including transient states and actions for both protocol variants. Further, we have modeled the complete protocol in the Murϕ model checker [17] and exhaustively verified the protocol for deadlock-freedom and safety, i.e., they enforce the Single-Write-Multiple-Reader invariant [66]. The detailed state transition table for the replica controller and the Murϕ model are available online[7].

5) Protocol Optimizations.: We describe 3 protocol optimizations for improving performance.

Speculative replica access. Consider an LLC miss that is sent to the replica directory in the allow-based protocol; further let us assume that an entry for the location is not present in the replica directory. This can either mean that the location is in writable state in one of home-LLCs or the directory entry has been evicted. The only way to find out for sure is to ask the home directory. But in the meantime the local replica can be speculatively accessed to overlap memory latency with home directory access. A similar optimization can be employed in the deny-based protocol as well: in case of a replica directory miss, the local replica can be speculatively accessed while waiting for the entry to be retrieved from DRAM.

Coarse-grained replica directory. Until now we had assumed that the replica directories operate at cache line granularity. We draw inspiration from prior works [8], [49] to amortize overheads by using coarse-grain replica directories. We use a contiguous, aligned block of memory to be covered by one entry when possible i.e. a full memory block is entered into the replica directory if no cacheline within it is currently in writable state.

Sampling based dynamic protocol. The performance of allow-based vs disallow-based protocols is dependent on the workload. A sampling based dynamic protocol can be used to achieve the best of both. We apply both approaches to a region of memory for a few epochs and monitor their effectiveness (similar in spirit to set dueling). Such an implementation requires a few additional saturating counters in the profiling phase. The scheme that performs better is then applied to the rest of the memory. When switching between the protocols (based on a register being set in the CPU by compiler or OS), we enter a drain phase to clear the replica directory and stop reading from the replica memory. We then switch state machines, followed by a warmup phase to bring the metadata entries *au courant*.

D. Discussion: OS support for memory replication

We now discuss the OS support needed to enable memory replication using the flexible table based mapping (relaxing the restriction of fixed function mapping assumed so far). There are 3 fundamental questions that need to be addressed for this: (i) How does the OS carve and manage space required for replication? (ii) How does the OS map replica page pairs? (iii) When does the OS enable or disable replication?

Carving and managing space required for replication: The OS already uses heuristics to estimate unused memory to transparently cache accessed files (called as disk buffering or file caching). Such approaches to estimate maximum DRAM

[7] https://github.com/adarshpatil/dve

resident set size can be reused to opportunistically steal system visible memory for replication. Further, balloon drivers [76] in the OS can be used to create memory pressure, forcing it to select pages to swap to disk, thereby carving memory space for replication. If memory pages are required to be swapped out to disk, the OS can monitor page fault rate to ensure that excessive swapping does not cause performance degradation beyond a pre-defined threshold. Note that Dvé only requires pairs of pages in different NUMA nodes and not a large contiguous address space, thus avoiding the need to perform memory compaction. In the absolute worst-case, when additional memory capacity is essential (either during burst periods or diurnal workloads), server management infrastructure can notify the OS to disable Dvé replication. The memory relinquished can be hot-plugged back to system visible capacity (free memory pool). Dvé's modular design of building over ECC enabled DIMMs with Chipkill allows the system to provide the baseline reliability when Dvé is disabled.

Mapping replica page pairs: As explained in Sec. III, replica page pairs are stored in an in-memory data structure called RMT. As the OS is already aware of the memory layout on boot via EFI it can create replica pairs such that they exist on memory adjoining different sockets. The RMT can be cached at the directory controller for quick lookups and the controllers can lookup/walk the RMT in hardware when needed (similar to a page table walker). Replication can even be performed at coarse granularity, allowing the RMT to be organized as a simple linear table or a 2-level radix-tree (similar to the page table) to perform fast end-to-end translations. A mapping entry can remain in the RMT despite the page being deallocated. This reduces the number of times the RMT cache would need to be shot down or quiesced. Further, RMT changes are infrequent since it only needs modifications in the rare event of a capacity crunch i.e., free list runs out, and the OS reclaims replica pages for use as addressable memory. Lastly, RMT entries can also be apriori populated by the OS heuristically in anticipation of replication requests for fast turn-around at memory allocation.

When should replication be enabled or disabled? The onus is on the workload placement and server management infrastructure (aka Control Plane) to define critical workloads and notify the OS when such replication costs are justified. The workload placement infrastructure manages the capacity vs reliability memory mode decision as a soft setting on a fleet of commodity high-capacity machines without having to procure specialized hardware for high reliability memory. This allows greater flexibility as datacenters and HPC installations can provision single homogeneous iso-config hardware to enable easier cost-efficient management while using Dvé to run mission critical workloads where reliability is a non-negotiable first order concern.

Dvé's replicated reliable memory can be flexibly deployed: (i) by the hypervisor at per-VM granularity, (ii) per-container or per-process granularity for serverless FaaS (Function-as-a-Service) deployments, (iii) for kernel allocations where system stability is of utmost importance or in workloads like file servers which use large amounts of OS memory to perform basic kernel operations, (iv) by mirroring entire address space to protect aging machines errors. With this knowledge, the OS adds a flag to the process control block (PCB) at creation process time to always allocate replicated memory.

Alternatively, to allow an application to explicitly provide high reliability to certain memory regions (say for a stateless application to allocate failure resilient data segments), a variant of the malloc/calloc call can be provided to request the OS to allocate a replicated physical memory.

Finally, note that Dvé guarantees higher reliability and improved performance only for the replicated region.

TABLE II
CONFIGURATION OF THE SIMULATED SYSTEM

Processors	16-core, 2 socket, (8-core/socket), 3.0 GHz
L1 I/D Cache	64KB, 8-way, private per core, 1 cycle, writeback
L2 Cache (LLC)	8MB, 16-way, shared per socket, 20 cycles, writeback
Local Directory	embedded in L2, fine-grain (cores) sharing vector
Global Directory	20-cycle, coarse-grain (sockets) sharing vector
Baseline Memory	2 × 8GB DDR4-2400 Mhz, 8 devices, 8-bit interface tCL-tRCD-tRP-tRAS=14.16ns-14.16ns-14.16ns-32ns 1KB row buffer, 16 banks/rank, 1 channel/socket
Replicated Memory	4 × 8GB DDR4-2400 Mhz, 2 channel/socket
Intra-socket interconnect	2×4 Mesh, SSSP routing, 1 cycle per hop
Inter-socket interconnect	point-to-point, 50ns

TABLE III
BENCHMARKS

Suite	Benchmark
HPC (assorted)	comd [1], xsbench [72], graph500 [21], rsbench [71]
PARSEC [5]	canneal, freqmine, streamcluster
SPLASH-2x [74]	barnes, fft, ocean_cp
Rodinia [11]	backprop, bfs, nw
NAS PB [53]	mg, bt, sp, lu
Parboil [34]	stencil, histo
SPEC 2017 [69]	lbm

E. Performance caveats of Dvé

Dvé aims to maximize the performance obtained by using the Coherent Replication scheme. Recall that if there are any issues with one of the replicas, for e.g. due to hard errors or thermal throttling or preventing a row-hammer access pattern, the system is placed in a degraded state where there is only one working copy. This negates the performance benefits for that memory location. Note that by marking the locations which are in a degraded state and funneling their requests to the single functional location, Dvé will provide performance comparable to baseline NUMA. The performance benefits may also be nullified or marginally adversely affected with Dvé if all compute and memory accessed is localized to a single NUMA node. This is because all memory writes have to be replicated to both NUMA nodes (not in critical path).

VI. EVALUATION METHODOLOGY

In this section, we set out the parameters and configuration of the multi-socket NUMA architecture used in our experimental evaluation of the coherent replication protocols proposed.

System Configuration: We model 2-socket Skylake-like configuration with mesh topology within the socket and a point-to-point QPI/UPI-like interconnect between the sockets. All

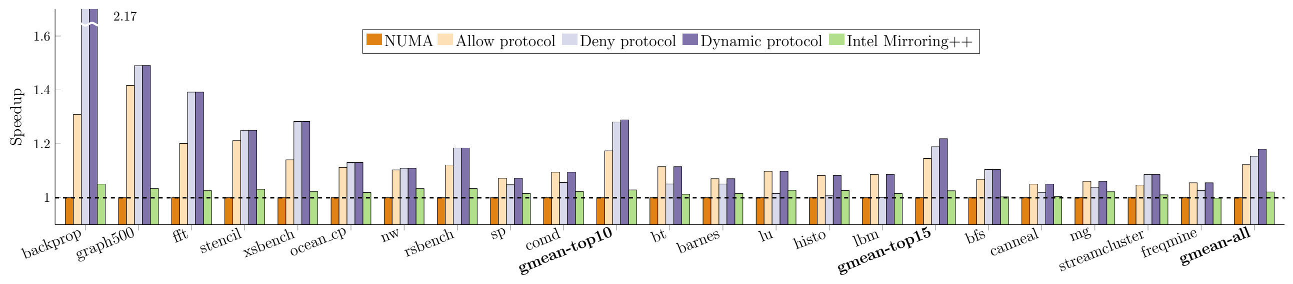

Fig. 6. Performance comparison of Dvé

links are ordered and have a fixed latency. A table-based static routing is enforced with a shortest path route with minimum number of link traversals. Each multi-core chip has per-core private caches, a shared LLC, a directory and a memory controller. The system is kept coherent using a hierarchical MOESI/MOSI protocol (full config details in Table II). Memory is allocated using an interleave policy whereby adjacent pages are interleaved across memory controllers in a round-robin fashion. We use an inter-socket latency of 50ns per hop. This is in-line with the difference between local and remote memory on a dual-socket Intel SandyBridge machine [23]. We also study the performance sensitivity to inter-socket latency.

Memory Configuration: Since our primary aim is to demonstrate the benefits of coherent replication in conditions where workloads memory needs are already satisfied, both the baseline and Dvé have the same system visible memory capacity. We assume the entire system memory is replicated using a fixed function mapping as explained in Sec III. In our evaluation, to accommodate the additional capacity required to store the replica, we add DIMMs on another channel on both the NUMA nodes as shown in Fig. 3. While several other lower performance configurations can be used to increase the capacity required to house the replica (like using a dual rank DIMM or higher capacity chips in a DIMM), this does not affect the overall reliability of the system as the replicas are anyway stored on different NUMA nodes.

Simulator: Simulating large core count systems with high-capacity DRAMs and large application working set sizes is challenging with existing publicly available tools. To circumvent this, we generate traces of benchmarks using the Prism framework [54] which uses Valgrind to generates traces of compute, memory, multi-threading APIs like create/join, mutexes, barriers, conditional wait/signaling/broadcasting, spin locks/unlocks and producer-consumer thread communication events. The tool captures synchronization and dependency-aware, architecture-agnostic multi-threaded traces.

We replay traces in a modified gem5 simulator [59]. Integer/floating point computations and thread API events have fixed latency of 1 cycle and 100 cycles respectively while all memory operations are simulated in detail. The replay mechanism respects synchronizations, barriers and mutexes. We skip the serial sections and/or initialization parts of the programs and instrument only the region of interest (ROI) in the benchmarks.

Workloads: We use OpenMP and Pthreads based multi-threaded workloads from 7 benchmark suites. Memory intensive applications were chosen from these suites (Table III). We use the largest input dataset available for these benchmarks. Traces of the ROI are used to warm-up the caches and structures for the first 1 billion operations i.e. computation, memory or communication events (which correspond to between 0.5-1.15 billion instructions) and then simulated in detail for 20 billion operations (8-19.3 billion instructions). We order the workloads in descending order of L2 MPKI and report the geometric mean of speedup as an aggregate statistic for the top-10 (high MPKI), top-15 and all 20 benchmarks.

Protocol Config: We use a fully associative 2K entry structure for the replica directory. We assume same access latency for the replica directory as the home directory for both protocols. We employ speculative replica memory access optimization as part of the default configuration. While this does trade-off bandwidth for squashed replies, we find that in our simulations the latency benefits outweigh the bandwidth loss. For the sampling based dynamic scheme optimization, we run a profile phase for the workload for 100 million instructions every 1 billion instructions for each scheme. We then apply the scheme that performs better for the rest of the phase.

VII. Evaluation Results

The goals of our evaluation are as follows. First, to evaluate the performance benefits of the coherent replication protocols over baseline NUMA architecture. Second, to explain the performance benefits using the key metric of coherence traffic between sockets. Third, evaluate the optimizations presented in Sec. V-C5. Fourth, evaluate the robustness of performance gains of Dvé schemes to varied inter-socket latencies. Finally, study the impact of replication on DRAM and system energy.

Performance Benefits of Coherent Replication: Fig. 6 shows the performance of allow, deny and dynamic protocol normalized to a baseline NUMA system without memory replication. The deny protocol achieves an average speedup of 28%, 18% and 15% for the top-10, top-15 and all benchmarks respectively, while the allow protocol achieves an average speedup of 17%, 14% and 12% respectively over the baseline. Although deny protocol performs better on average, only 10 benchmarks (backprop, graph, fft, stencil, xsbench, ocean_cp, nw, rsbench, bfs, streamcluster) show better performance with a deny protocol while the other workloads perform better with the allow protocol. The dynamic protocol is always able to detect the better of 2 protocols. Therefore the dynamic protocol achieves the best average speedup of 29%, 22% and 18% over the baseline. A key point to note is that all benchmarks for all schemes, perform equal to or better than the baseline which shows that the overheads of coherent replication does not cause any adverse performance penalties.

To compare against the best possible performance that can be gained by reading data from 2 mirrored channels, we implement an improved (hypothetical) version of Intel's memory mirroring scheme by actively load balancing reads between them. (Recall the default Intel memory mirroring scheme does not read the secondary copy, unless the primary fails.) Dvé's schemes provide a geomean speedup of 9% and 13% better than the Intel-mirroring++ scheme for allow and deny respectively, as they are able to avoid the inter-socket latency by providing node local reads while the Intel-mirroring++ scheme can only provide higher read memory bandwidth by allowing reads from both channels.

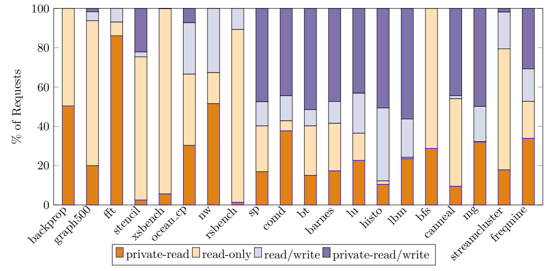

Fig. 7. Sharing pattern in benchmarks

Performance Analysis - Workload Sharing Characteristics: To assess why a scheme performs better for a workload, we look at the inter-socket sharing characteristics of workloads in the baseline NUMA system. By analyzing this, we can understand the opportunity for performance gains from each scheme. We classify the requests at the home directory as one of the following – *private-read* for a GETS request to a line in I state, *read-only* for a GETS request to a line in S state, *read/write* for a GETS request to a line in M or O state or a GETX request to a line in S state, *private-read/write* for GETX request to a line in I state. Fig. 7 shows the distribution of the above mentioned classes for each workload. Workloads that exhibit considerable private read/write behavior (greater than 46%) show higher benefits with an allow protocol. This is expected, since for such workloads on a GETX request where there is no sharing, the allow protocol is able to avoid aggressively pushing invalidates (by virtue of being a lazy, pull-based scheme) while the deny protocol is required to update the replica directory.

Performance Analysis - Inter-socket traffic: Fig. 8 quantifies the reduction in inter-socket traffic when using Dvé protocols compared to baseline NUMA. We see that reduction in inter-socket traffic correlates with the performance benefits, i.e., the protocol which shows higher reduction in inter-socket traffic

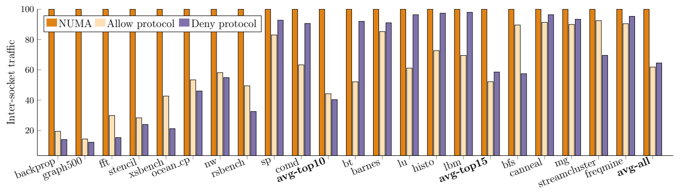

Fig. 8. Inter-socket traffic (normalized to NUMA)

performs better. Backprop and graph500 notably experience a 86% and 84% reduction in inter-socket traffic as they see mostly private-read and read-only requests which can be serviced by the replica. Overall for the 20 benchmarks, allow and deny protocols reduce inter-socket communication traffic by an average of 38% and 35% respectively over the baseline NUMA architecture.

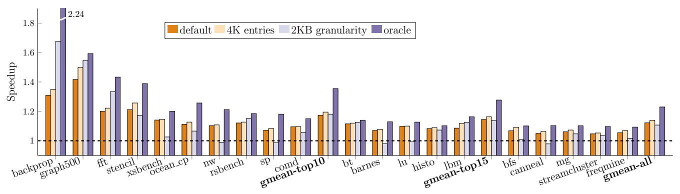

Fig. 9. Allow-based protocol optimizations

Optimizing the schemes: To gauge the ceiling of performance we can expect from an allow-based protocol, we measure the performance achievable with an oracular allow-based scheme. Intuitively, the size of replica directory structure is proportional to the efficiency of the scheme and thus, for an oracular scheme we allow the replica directory entries to be infinite and exhaustive. Further, it does not incur any latency overheads to add an entry (in the spirit of oracle knowledge) while invalidation/removal of entries does incur latency (since this can never be avoided). With such a configuration, we see that the performance of oracular scheme is 18.3% and 10.8% better than the default allow protocol for the top10 and all benchmarks respectively (Fig. 9, fourth bar).

In reality to achieve near oracular performance, we double the number of replica directory entries to a fully associative 4K entry structure. We see that the larger structure provides an average hit rate improvement of 32% and improves performance by 2.1% and 1.7% over the default allow protocol for top10 and all benchmarks respectively (Fig. 9, second bar).

Another optimization we can use is to use coarse-grain tracking at the replica directory to increase the reach. When such an optimization is employed benchmarks such as backprop, graph500, fft, rsbench show gains while stencil, ocean_cp, comd, bfs suffer compared to the default cache-line level tracking (Fig. 9, third bar). Further nw, sp, barnes and canneal perform worse than baseline NUMA. This is due to the additional overhead to invalidate entries from other sockets when an exclusive request is issued for a larger granularity. Overall, coarse granularity tracking performs 0.7% better for top10 benchmarks but performs worse by 1.7% over all benchmarks compared to the default allow protocol. Thus such a technique is not well suited to improve performance.

We note that the performance of the best variant of the allow protocol for each benchmark is within 12% and 7% of the oracular performance for top10 and all benchmarks (16% better than the baseline NUMA overall).

Sensitivity to inter-socket latency: A number of software-based techniques – including Carrefour [16], Shoal [35], AutoNUMA – have been proposed to mitigate NUMA effects; the net effect of each of these techniques is to reduce the average inter-socket latency. Therefore, we study the effect of inter-socket interconnect latencies on the performance of Dvé. As seen in Fig. 10, even with fairly low 30ns interconnect latency (each way), the deny protocol outperforms the baseline by 19%, 12% and 10% for top-10, top-15 and all benchmarks respectively. On the other end, with increased inter-socket latencies of 60ns (as in emerging scalable long range interconnects like CCIX [9], OpenCAPI [55] and GenZ [19]), Dvé's benefits increase as expected.

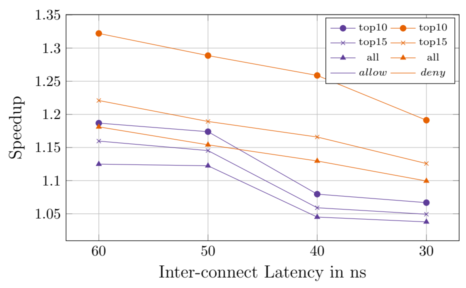

Fig. 10. Sensitivity to inter-connect latency

Energy: To understand the energy overheads of maintaining a replica for each location, we measure the energy-delay product (EDP) of the DRAM subsystem using the Micron datasheet [46]. We observe that, memory-EDP for memory intensive benchmarks (backprop, graph500, fft) reduces even with the overheads of double memory capacity but the geomean over all benchmarks increases by 43% and 37% for allow and deny protocols respectively. (Note that we expect the memory-EDP of Dvé to be even lower when using idle memory as it still uses energy for refresh, even in a low power (self-refresh) state.) Moreover, typically memory consumes only a fraction of the overall system power: about 18% of the total system power in a 2-socket NUMA system [2]. Using this to calculate the system-EDP, we find that the system-EDP geomean over all benchmarks turns out to be lower by 6% and 12% for allow and deny respectively, due to shorter execution times.

VIII. Related Work

1) *DRAM RAS Proposals*:

State-of-the-art ECC proposals: A large body of recent work have proposed several variants of ECC schemes to deal with errors in the DRAM devices [10], [20], [32], [36], [37], [51], [73], [79]. Specifically, multi-tier ECC approaches proposed in [31], [73], [79] separate error detection and error correction code. AIECC [37] sheds light on the need for channel error protection (clock, control, command and address buses) and a holistic ECC based scheme to achieve it.

All these proposals rely on correction bits stored in the same DIMM/rank/channel and thus cannot correct memory controller or channel failures as in this work. Dvé can be flexibly paired with any proposed ECC scheme for error detection, and use the replica to recover from an error. These works also suffer performance degradation for providing additional protection while Dvé provides performance benefits.

Non-ECC based DRAM RAS schemes: To detect transmission errors on channels, DDR4 memory controllers use CRCs and retry transactions if errors are detected [46]. If errors persist, lanes are quiesced, reset and recalibrated but these cannot handle hard channel failures. Aside from ECC based systems, Intel [22], [26] and IBM processors [25] allow memory to be mirrored across two channels within a memory controller or across two DIMMs within a channel. While these mechanisms can tolerate channel failures, they are still subject to faults in the single memory controller subsystem.

MemGuard [13] uses incremental log hashes for error detection and a OS created checkpoint for error recovery. However the recovery is not instant and can lead to loss of updates. Selective word level replication [52] proposes selective replication of words to mitigate large granularity failures in DRAM chips with manufacturing defects. As before, all these mechanisms cause performance degradation for providing at most channel error protection.

Stacked DRAM RAS schemes: Stacked DRAMs are prone to failures in dies and TSVs due to their organization, which resemble chip and channel failures. Adopting conventional ECC schemes causes performance pathologies in stacked DRAMs due to the layout of data. To protect against such errors without causing performance/power overheads a body of work [12], [29], [33], [43], [50] calls for adding additional tier-2 code that is an XOR of the data blocks. This XOR block is then stored in the same stacked DRAM in a manner that allows regeneration of original block in case of a TSV/die failure. Although the XOR block is stored in an independent channel it cannot be accessed independently without the data block (as it is not a full replica). Even if a replica were used instead of the XOR block, these works would resemble at best an Intel mirroring-like approach [26].

2) *Mitigating NUMA overheads:*

Carrefour [16] proposes OS-driven selective replication of memory read-only or read-mostly pages to alleviate NUMA overheads. Shoal [35] proposed program analysis to automatically replicate memory regions across NUMA nodes for mitigating the performance penalty of remote access.

Architectural solutions for mitigating NUMA has had a long history, most notably in the cache-only-memory architecture (COMA) and tertiary caching line of research [3], [61], [75], [81]. More recently, C3D [23] and Candy [15] use a per-socket stacked DRAM cache to reduce NUMA interconnect latencies. All of these works leverage hardware support for caching remote data and keeping them consistent. None of the aforementioned works aim to provide fault-tolerance. While

Dvé aims to use the replica for improving performance, it's primary goal is to provide improved DRAM fault-tolerance.

*3) **NVM RAS mechanisms**:* NVMs suffer from high number of hard errors due to cell wear out and stuck-at faults. ECP [63], FREE-p [78], PAYG [57] and Chipkill-correct for NVRAM [80] propose using pointers or error correction entries to remap failed bits in a line or use variants of ECC. All these schemes still target cell failures and cannot deal with larger granularity failures due to errors in shared components.

Dynamically Replicated Memory (DRM) [27] uses a read-after-write scheme to detect errors. When an error is detected, the scheme makes a copy to a compatible page with dissimilar faults. Both copies need to be accessed to service a request. We find that NVM memory RAS mechanisms are also insufficient to handle the scope of errors targeted in our work.

*4) **Exploiting underutilized memory**:* FMR [56] proposes replicating data into idle memory – on different ranks in the same memory controller and uses it to hide DRAM maintenance latency overheads like refresh, precharge delay, etc. FMR performs lock step write into the rank replicas within the same memory controller. This is very similar to the Intel-mirroring++ scheme implemented in our evaluation.

Workload co-location exploiting workload heterogeneity and variability has been proposed to improve utilization in the cloud [42]. However, this is having limited effectiveness given the dynamic bursty nature coupled with stringent tail latency, throughput guarantee requirements and increased security concerns. Memory disaggregation has also been proposed to allow workloads that require high capacity memory to take advantage of idle remote memory to improve performance [28], which is orthogonal to our approach.

IX. Conclusion

We have presented Dvé, a hardware memory replication mechanism for achieving both reliability and performance. We have demonstrated that this unique design point offers considerably higher memory reliability and can be flexibly deployed on-demand. Furthermore, our experimental results indicate that in contrast to existing memory reliability techniques that are detrimental to performance, Dvé provides a non-trivial improvement in performance.

X. Acknowledgements

We thank Nikos Nikoleris, Dan Sorin, Vasilis Gavrielatos, Jayvant Anantpur and the anonymous shepherd/reviewers whose feedback helped to refine our ideas and improved the paper. This work was supported in parts by ARM, through their PhD Scholarship Program and NSF grant CNS-1718834.

References

[1] ARM HPC, "arm-hpc/comd," https://github.com/arm-hpc/CoMD.

[2] L. A. Barroso, U. Hölzle, P. Ranganathan, and M. Martonosi, *The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition*. Morgan & Claypool Publishers, 2018.

[3] S. Basu and J. Torrellas, "Enhancing memory use in simple coma: Multiplexed simple coma," in *HPCA*, 1998.

[4] L. Bautista-Gomez, F. Zyulkyarov, O. Unsal, and S. McIntosh-Smith, "Unprotected computing: A large-scale study of dram raw error rate on a supercomputer," in *SC*, 2016.

[5] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[6] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[7] S. Blanas, "Near data computing from a database systems perspective," https://www.sigarch.org/near-data-computing-from-a-database-systems-perspective.

[8] J. F. Cantin, M. H. Lipasti, and J. E. Smith, "Improving multiprocessor performance with coarse-grain coherence tracking," in *ISCA*, 2005.

[9] CCIX consortium, http://www.ccixconsortium.com.

[10] S. Cha, O. Seongil, H. Shin, S. Hwang, K. Park, S. J. Jang, J. S. Choi, G. Y. Jin, Y. H. Son, H. Cho, J. H. Ahn, and N. S. Kim, "Defect analysis and cost-effective resilience architecture for future dram devices," in *HPCA*, 2017.

[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2009.

[12] H.-M. Chen, C.-J. Wu, T. Mudge, and C. Chakrabarti, "Ratt-ecc: Rate adaptive two-tiered error correction codes for reliable 3d die-stacked memory," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 3, Sep. 2016.

[13] L. Chen and Z. Zhang, "Memguard: A low cost and energy efficient design to support and enhance memory system reliability," in *ISCA*, 2014.

[14] W. Chen, K. Ye, Y. Wang, G. Xu, and C. Xu, "How does the workload look like in production cloud? analysis and clustering of workloads on alibaba cluster trace," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018.

[15] C. Chou, A. Jaleel, and M. K. Qureshi, "Candy: Enabling coherent dram caches for multi-node systems," in *MICRO*, 2016.

[16] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quema, and M. Roth, "Traffic management: A holistic approach to memory placement on numa systems," in *ASPLOS*, 2013.

[17] D. L. Dill, "The murphi verification system," in *Proceedings of the 8th International Conference on Computer Aided Verification*, ser. CAV '96. Berlin, Heidelberg: Springer-Verlag, 1996.

[18] M. Fieback, "Dram reliability: Aging analysis and reliability prediction model," https://repository.tudelft.nl/islandora/object/uuid:e36c2de7-a8d3-4dfa-9da1-ac5b7e18614b, 2017.

[19] Gen-Z consortium, http://genzconsortium.org.

[20] S. Gong, J. Kim, S. Lym, M. Sullivan, H. David, and M. Erez, "Duo: Exposing on-chip redundancy to rank-level ecc for high reliability," in *HPCA*, 2018.

[21] Graph500, "github/graph500," https://github.com/graph500/graph500.

[22] HP, "Advanced memory protection technologies, technology brief, 5th edition," ftp://ftp.hp.com/pub/c-products/servers/options/c00256943.pdf.

[23] C. Huang, R. Kumar, M. Elver, B. Grot, and V. Nagarajan, "C3d: Mitigating the numa bottleneck via coherent dram caches," in *MICRO*, 2016.

[24] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: Understanding the nature of dram errors and the implications for system design," in *ASPLOS*, 2012.

[25] IBM, "Power processor-based systems ras, june 27th, 2019," https://www.ibm.com/downloads/cas/2RJYYJML.

[26] Intel, "Address range partial memory mirroring," https://software.intel.com/content/www/us/en/develop/articles/address-range-partial-memory-mirroring.html and https://01.org/lkp/blogs/tonyluck/2016/address-range-partial-memory-mirroring-linux.

[27] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically replicated memory: Building reliable systems from nanoscale resistive memories," in *ASPLOS*, 2010.

[28] M. G. Ivy Peng, Roger Pearce, "On the memory underutilization: Exploring disaggregated memory on hpc systems," in *2020 32st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2020.

[29] H. Jeon, G. H. Loh, and M. Annavaram, "Efficient ras support for die-stacked dram," in *2014 International Test Conference*, 2014.

[30] X. Jian, N. DeBardeleben, S. Blanchard, V. Sridharan, and R. Kumar, "Analyzing reliability of memory sub-systems with double-chipkill detect/correct," in *2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing*, 2013.

[31] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar, "Low-power, low-storage-overhead chipkill correct via multi-line error correction," in *SC*, 2013.

[32] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," in *HPCA*, 2013.
[33] X. Jian, V. Sridharan, and R. Kumar, "Parity helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems," in *HPCA*, 2016.
[34] I.-J. S. John A. Stratton, Christopher Rodrigues, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W. mei W. Hwu, "Parboil: A revised benchmark suite for scientificand commercial throughput computing," *IMPACT-12-01*, March 2012.
[35] S. Kaestle, R. Achermann, T. Roscoe, and T. Harris, "Shoal: Smart allocation and replication of memory for parallel programs," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015.
[36] J. Kim, M. Sullivan, and M. Erez, "Bamboo ecc: Strong, safe, and flexible codes for reliable computer memory," in *HPCA*, 2015.
[37] J. Kim, M. Sullivan, S. Lym, and M. Erez, "All-inclusive ecc: Thorough end-to-end protection for reliable computer memory," in *ISCA*, 2016.
[38] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *ISCA*, 2014.
[39] L. A. Lastras-Montaño, P. J. Meaney, E. Stephens, B. M. Trager, J. O'Connor, and L. C. Alves, "A new class of array codes for memory storage," in *2011 Information Theory and Applications Workshop*, 2011.
[40] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-latency dram: Optimizing dram timing for the common-case," in *HPCA*, 2015.
[41] S. Liu, B. Leung, A. Neckar, S. O. Memik, G. Memik, and N. Hardavellas, "Hardware/software techniques for dram thermal management," in *HPCA*, 2011.
[42] C. Lu, K. Ye, G. Xu, C. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2884–2892.
[43] G. Mappouras, A. Vahid, R. Calderbank, D. R. Hower, and D. J. Sorin, "Jenga: Efficient fault tolerance for stacked dram," in *ICCD*, 2017.
[44] P. J. Meaney, L. A. Lastras-Montano, V. K. Papazova, E. Stephens, J. S. Johnson, L. C. Alves, J. A. O'Connor, and W. J. Clarke, "Ibm zenterprise redundant array of independent memory subsystem," *IBM Journal of Research and Development*, vol. 56, no. 1.2, Jan 2012.
[45] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2015.
[46] Micron, "DDR4 SDRAM Datasheet," https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/8gb_ddr4_sdram.pdf.
[47] Micron, "DDR5 SDRAM Whitepaper," https://www.micron.com/-/media/client/global/documents/products/white-paper/ddr5_more_than_a_generational_update_wp.pdf.
[48] Micron, "Technical note: Uprating semiconductors for high-temperature applications," http://notes-application.abcelectronique.com/024/24-20035.pdf.
[49] A. Moshovos, "Regionscout: exploiting coarse grain sharing in snoop-based coherence," in *ISCA*, 2005.
[50] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "Citadel: Efficiently protecting stacked memory from large granularity failures," in *MICRO*, 2014.
[51] P. J. Nair, V. Sridharan, and M. K. Qureshi, "Xed: Exposing on-die error detection information for strong memory reliability," in *ISCA*, 2016.
[52] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "Archshield: Architectural framework for assisting dram scaling by tolerating high error rates," in *ISCA*, 2013.
[53] NASA Advanced Supercomputing Division, "Nas parallel benchmarks," https://www.nas.nasa.gov/publications/npb.html.
[54] S. Nilakantan, K. Sangaiah, A. More, G. Salvadory, B. Taskin, and M. Hempstead, "Synchrotrace: synchronization-aware architecture-agnostic traces for light-weight multicore simulation," in *ISPASS*, 2015.
[55] OpenCAPI consortium, http://opencapi.org.
[56] G. Panwar, D. Zhang, Y. Pang, M. Dahshan, N. DeBardeleben, B. Ravindran, and X. Jian, "Quantifying memory underutilization in hpc systems and using it to improve performance via architecture support," in *MICRO*, 2019.
[57] M. K. Qureshi, "Pay-as-you-go: Low-overhead hard-error correction for phase change memories," in *MICRO*, 2011.
[58] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing (SoCC)*, 2012.
[59] K. Sangaiah, M. Lui, R. Jagtap, S. Diestelhorst, S. Nilakantan, A. More, B. Taskin, and M. Hempstead, "Synchrotrace: Synchronization-aware architecture-agnostic traces for lightweight multicore simulation of cmp and hpc workloads," *ACM Trans. Archit. Code Optim.*, Mar. 2018.
[60] Sanghyuk Kwon, Young Hoon Son, and Jung Ho Ahn, "Understanding ddr4 in pursuit of in-dram ecc," in *2014 International SoC Design Conference (ISOCC)*, 2014.
[61] A. Saulsbury, T. Wilkinson, J. Carter, and A. Landin, "An argument for simple coma," in *HPCA*, 1995.
[62] T. Scharon Harding, "Ecc memory in dram," https://www.tomshardware.com/uk/reviews/ecc-memory-ram-glossary-definition,6013.html.
[63] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ecp, not ecc, for hard failures in resistive memories," in *ISCA*, 2010.
[64] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: A large-scale field study," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, Jun. 2009.
[65] T. Siddiqua, A. E. Papathanasiou, A. Biswas, S. Gurumurthi, I. Corp, and T. Aster, "Analysis and modeling of memory errors from large-scale field data collection," in *SELSE*, 2013.
[66] D. J. Sorin, M. D. Hill, and D. A. Wood, *A Primer on Memory Consistency and Cache Coherence*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2011.
[67] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *SC*, 2012.
[68] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad, and the ugly," in *ASPLOS*, 2015.
[69] Standard Performance Evaluation Corporation, "SPEC CPU 2017," https://www.spec.org/cpu2017/.
[70] M. Taassori, R. Balasubramonian, S. Chhabra, A. R. Alameldeen, M. Peddireddy, R. Agarwal, and R. Stutsman, "Compact leakage-free support for integrity and reliability," in *ISCA*, 2020.
[71] J. R. Tramm, A. R. Siegel, B. Forget, and C. Josey, "Performance analysis of a reduced data movement algorithm for neutron cross section data in monte carlo simulations," in *EASC 2014 - Solving Software Challenges for Exascale*, Stockholm, 2014.
[72] J. R. Tramm, A. R. Siegel, T. Islam, and M. Schulz, "XSBench - the development and verification of a performance abstraction for Monte Carlo reactor analysis," in *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*, Kyoto, 2014.
[73] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi, "Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems," in *ISCA*, 2012.
[74] P. G. P. University, "A memo on exploration of splash-2 input sets," https://parsec.cs.princeton.edu/doc/memo-splash2x-input.pdf.
[75] B. Verghese, S. Devine, A. Gupta, and M. Rosenblum, "Operating system support for improving data locality on cc-numa compute servers," in *ASPLOS*, 1996.
[76] C. A. Waldspurger, "Memory resource management in vmware esx server," *SIGOPS Oper. Syst. Rev.*, p. 181–194, Dec. 2003.
[77] R. Yeleswarapu and A. K. Somani, "Sscmsd - single-symbol correction multi-symbol detection for dram subsystem," in *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2018.
[78] D. H. Yoon, N. Muralimanohar, J. Chang, P. Ranganathan, N. P. Jouppi, and M. Erez, "Free-p: Protecting non-volatile memory against both hard and soft errors," in *HPCA*, 2011.
[79] D. H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in *ASPLOS*, 2010.
[80] D. Zhang, V. Sridharan, and X. Jian, "Exploring and optimizing chipkill-correct for persistent memory based on high-density nvrams," in *MICRO*, 2018.
[81] Z. Zhang and J. Torrellas, "Reducing remote conflict misses: Numa with remote cache versus coma," in *HPCA*, 1997.
[82] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-rank: Adaptive dram architecture for improving memory power efficiency," in *MICRO*, 2008.