Disintegrating Manycores: Which Applications Lose and Why?

Isidor R. Brkić
igi.brkic@mail.utoronto.ca
University of Toronto
Canada

Mark C. Jeffrey
mcj@ece.utoronto.ca
University of Toronto
Canada

ABSTRACT

The economics of Moore's Law are stumbling, so vendors of manycore architectures are transitioning from single-die *monolithic* designs to multi-chiplet *disintegrated* systems within a package. Disintegration lowers cost for the same number of cores but bottlenecks the interconnect. Ideally, disintegration should increase performance per dollar: cost savings should outweigh the *disintegration slowdown*. Although industry has reported cost savings, the performance penalty of disintegration is not well studied.

This paper presents the first characterization, to our knowledge, of disintegration performance penalty across a diverse suite of applications. Unsurprisingly, applications with high speedups on monolithic systems continue to scale well on disintegrated systems, and vice versa. However, the disintegration slowdown compared to an equivalently sized monolith exhibits high variance across applications, with some achieving just over half the performance. Why do some applications get a performance per dollar win, while others lose? Through regression analysis, we find that metrics relating to the network-on-package bandwidth and data sharing correlate with disintegration slowdown. Programmers were already cautioned against shared mutable data on monolithic systems, yet data sharing is unavoidable in many applications. These applications will be disproportionately harmed in the disintegrated future.

CCS CONCEPTS

• **Networks → Network on chip**; • **Computer systems organization → Multicore architectures**.

KEYWORDS

silicon interposer, multi-chip module, disintegration slowdown, chiplets, data sharing, regression

ACM Reference Format:
Isidor R. Brkić and Mark C. Jeffrey. 2023. Disintegrating Manycores: Which Applications Lose and Why?. In *16th International Workshop on Network on Chip Architectures (NoCArc '23), October 28, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3610396.3618090

1 INTRODUCTION

Moore's Law is slowing, but user demand for higher performance continues unabated. Absent transistor scaling, performance comes from increased area for more parallelism or larger structures. However, increasing chip area is not a free lunch for two reasons: yield and manufacturing limits. As the die size increases, the probability of defects rises significantly, lowering manufacturing yield and increasing cost. Furthermore, standard manufacturing techniques have a maximum manufacturable die size due to limits of the lithographic reticle [27]. Even keeping the die size constant becomes more costly in newer technology nodes as increasing transistor density requires increasingly complex techniques [31].

This constraint on die area has led industry to re-integrate multiple chiplets within a package into so-called *disintegrated* [11, 24] systems. They fall between integrating all logic on a single chip and the decades-old multi-socket systems on a board. Chiplets communicate through silicon interposers [24], the package substrate [31], or other technologies [32]. Disintegrated systems are rapidly emerging in products from AMD [17, 25, 31], Apple [4], IBM [18], Intel [32], and Tenstorrent [29]. Disintegration allows vendors to build systems that exceed the maximum conventional manufacturable size, and replace infeasibly expensive monolithic chips with practical systems due to higher yield of smaller chips.

An underlying assumption has been that disintegrated systems provide substantially lower cost for a *small* performance loss. After all, although an in-package interconnect is less performant than one on-chip, it beats connecting packages on a board. While the cost savings are readily apparent (e.g., AMD estimates 41% cost reduction in the design of a 32-core system [31]), there has been little work characterizing the performance loss.

Our goal is to understand the performance tradeoff in the transition to disintegrated systems, identifying the properties of applications that cause them to be more or less impacted. We compare the performance of a diverse suite of 29 multithreaded applications running on two 256-core disintegrated systems (active interposer and multi-chip module) to a monolith with the same number of cores and amount of cache (Sec. 3). We introduce *disintegration slowdown*, the performance ratio of the disintegrated system and the equivalently sized monolith. We find that applications that scale well on monolithic systems continue to scale on the disintegrated systems, and vice versa, but provide 0.91× and 0.72× the average performance of the monolith, respectively (Sec. 2.2). However, these average disintegration slowdowns do not tell the whole story, as we observe significant variance across applications. For example, an application's speedup on the monolithic 256-core system over a 1-core system is a poor predictor of disintegration slowdown: some applications with monolithic speedup >250× have worse disintegration slowdown (0.56×) than some with monolithic speedup <100× (0.997×). We apply regression analysis in search of metrics that predict disintegration slowdown, identifying that instructions per invalidation, our data-sharing proxy for operational intensity, correlates with $R^2 \approx 0.6$ (Sec. 4).

NoCArc '23, October 28, 2023, Toronto, ON, Canada
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *16th International Workshop on Network on Chip Architectures (NoCArc '23), October 28, 2023, Toronto, ON, Canada*, https://doi.org/10.1145/3610396.3618090.

2 MOTIVATION

Disintegrating a manycore into chiplets drives down manufacturing cost, with the ultimate goal of improving performance per dollar.[1] Ideally, for any application, A,

$$\frac{P_d(A)}{C_d} > \frac{P_m(A)}{C_m} \iff \frac{P_d(A)}{P_m(A)} > \frac{C_d}{C_m} \tag{1}$$

where C_m and C_d are the fixed cost per manycore using monolithic and disintegrated designs, respectively, and $P_m(A)$ and $P_d(A)$ are the performance running A on the monolithic and disintegrated systems, respectively. The relative disintegrated cost should decrease more than relative disintegrated performance is harmed.

When does disintegration pay off and why do some applications lose? We call the left hand side, $P_d(A)/P_m(A)$, *disintegration slowdown*. For a given disintegrated and monolithic system, the relative cost (right hand side) is fixed (e.g., 0.59× for the 32-core, 4-chiplet AMD EPYC [31]). However, the disintegration slowdown depends not only on the fixed disintegrated and monolithic architectures, but also on the application, A. Prior work on disintegration has *(i)* summarized disintegration slowdown averaged across a suite of applications [6], *(ii)* measured disintegration slowdown on applications amenable to data partitioning to reduce communication [36], or *(iii)* has not measured disintegration slowdown in particular [7, 10, 12, 24, 31, 32, 35, 42, 48, 49, 50]. To the best of our knowledge, no prior work has characterized the *variance* in performance penalty of disintegration across applications from disparate domains, nor characterized its cause. The following subsections provide brief background on disintegration technology and motivate this work by showing the high variance of disintegration slowdown.

2.1 Disintegration (variably) reduces cost

Fig. 2 shows our baseline monolithic tiled, cache-coherent manycore with 256 cores. Each tile has a group of cores with private L1 caches. The cores in a tile share an L2 cache, and each tile has a slice of a fully shared L3 cache. Tiles are connected through a mesh network-on-chip (NoC). Even with simple and small cores, a system this large or larger would be costly to manufacture as chip yield decays superlinearly with area [13].

Disintegrated systems reintegrate chiplets to reduce overall cost. First, they reduce manufacturing cost as the significant yield improvement with lower area allows assembling a large system with smaller, cheaper chips [44]. Second, they amortize design cost: just as the shift to single-chip multiprocessors made large families of

[1]Other objectives include performance/W/$ or applying utility functions to performance [37], but are out of scope for this paper.

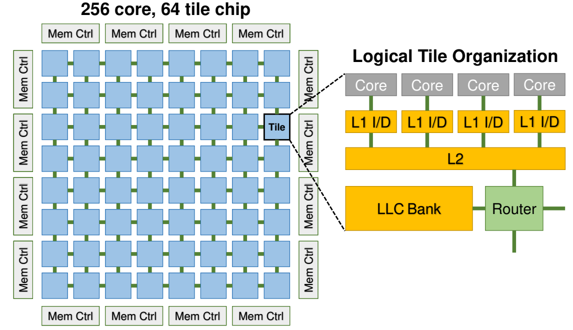
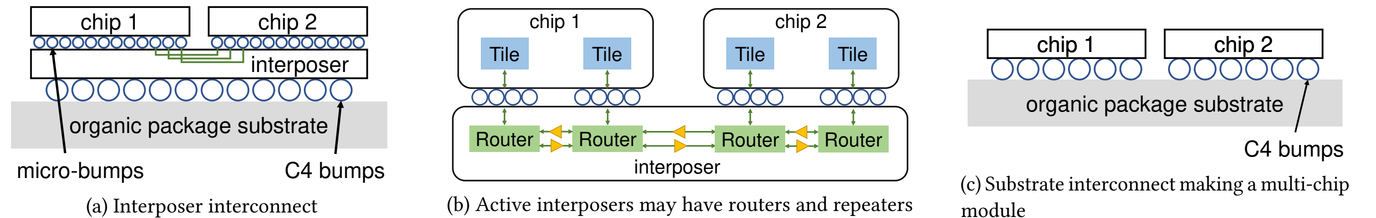


Figure 2: Tiled 256-core monolithic manycore design with three-level cache hierarchy

related processors economically feasible by replicating a core design [33], disintegrated systems enable populating a package with two, four, eight, or more chips to create a diverse product line from few chip designs [31].

Disintegrated systems require a means to reassemble and interconnect the chips within a package. The more integrated the technology the better the inter-chip bandwidth, latency, and energy [44] but the higher the cost. These include 3D stacking, interposers, silicon bridge interconnects, and multi-chip modules (MCM). We defer to prior work for summaries of this large technology space [10, 14, 24]. In this paper, we sample two representative points in the space with distinct cost and bandwidth/latency properties, focusing on active silicon interposers and MCMs. The latter are already deployed commercially in AMD EPYC and Ryzen [25, 31] and will likely connect chips in the Tenstorrent Grendel [29]. The former have so far been of academic interest, but represent a high-performance design point in the Universal Chiplet Interconnect (UCIe) standard [14] with support from AMD, IBM, Intel, among other vendors.

Active Silicon Interposers: Fig. 1a illustrates a disintegrated system connecting two chips through a silicon interposer. Although the interposer area is comparable to the sum of chiplet area, it only supports communication, making its active area a small fraction to reduce the probability of defects and improve yield. At a high level, there are two types of interposers: active (with powered transistors and logic) and passive (only metal wires). Fig. 1b zooms in on the chips and an active silicon interposer, which includes repeaters to reduce latency [12, 42] and routers to enable sophisticated network topologies [24]. In contrast, passive interposers have worse communication characteristics but achieve higher yield. A chip-to-interposer interconnect enables so-called micro-bump



(a) Interposer interconnect (b) Active interposers may have routers and repeaters (c) Substrate interconnect making a multi-chip module

Figure 1: Cross-section of on-package interconnects for two chiplets

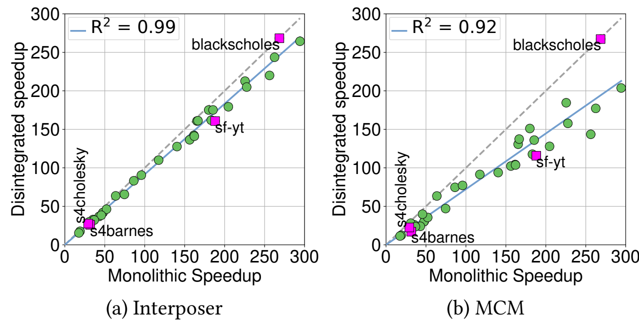
connections whereas chip-to-package reintegration requires C4 bumps. Micro-bumps are smaller and more densely packed than C4 bumps (e.g., 9× [24]) leading to higher inter-chip bandwidth through the interposer than through the package substrate.

Multi-Chip Modules: Fig. 1c shows a system connecting two chips through the package substrate. MCMs are cheaper than interposer-based systems as they demand fewer silicon layers. However, the package substrate does not support active logic such as routers or repeaters. The need for C4 bumps limits wire density and bandwidth. The lack of repeaters drives up inter-chip long-wire latency.

2.2 Disintegration (variably) hurts performance

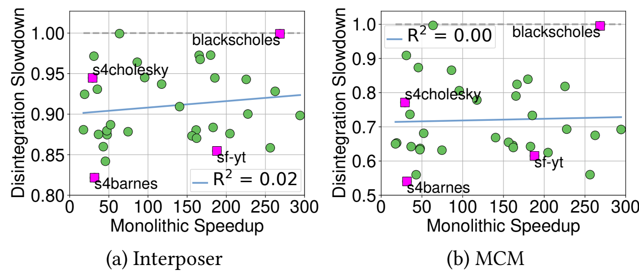
Keeping cores and cache equal with a monolith, disintegration improves *peak* performance per dollar by keeping peak performance constant, but what is the actual performance loss and how does it vary across applications? Fig. 3 compares the performance of a monolithic system (x-axis) against a disintegrated system (y-axis) with 256 cores across 34 application-input pairs (see Sec. 3 for methodology). Each point in the scatter plot represents the speedups normalized to the performance of a 1-core system, averaged over 10 runs of a benchmark with the same input. The x-axis values (monolith) are the same in Fig. 3a and Fig. 3b, while the y-axis values come from an interposer- and MCM-based system, respectively.

Unsurprisingly, disintegration hurts performance on average, shown by the blue least-squares-error lines of best fit. We force these lines through the origin. The coefficient of determination (R^2)



(a) Interposer (b) MCM

Figure 3: Comparing disintegrated vs. monolithic performance on a 256-core system normalized to a 1-core system



(a) Interposer (b) MCM

Figure 4: Comparing disintegration slowdown vs. monolithic performance on a 256-core system

shows a strong linear correlation between disintegrated speedup and monolithic speedup (R^2 =1 implies perfect correlation and R^2 =0 implies no correlation). The slopes indicate that interposer and MCM systems achieve 0.91× and 0.72× the performance of the monolith, respectively. This average relative performance is promising if disintegration sufficiently reduces cost.

More interestingly, monolithic speedup does not perfectly predict disintegrated speedup for individual applications. Fig. 4 highlights this issue, keeping the x-axis the same but plotting what we call disintegration slowdown on the y-axis: disintegrated speedup normalized to monolithic speedup, or $P_d(A)/P_m(A)$ from Eq. 1. For example, a y-axis value of one indicates a benchmark took equal time to execute on both systems, whereas a value of 0.5 indicates the disintegrated system took twice as long to complete.

Fig. 4 shows high variance in disintegration slowdown across applications, with near-zero correlation with monolithic speedup and near-zero slope in the line of best fit. For example, applications `blackscholes` and `sf` both scale well (to the right) on the monolith but their disintegration slowdown varies wildly. At the lower end of scalability, `barnes` is heavily penalized on the MCM, whereas `cholesky` is closer to the average disintegration slowdown. The standard deviations in disintegration slowdown are 0.046× and 0.116× for the interposer and MCM, respectively.

Our goal in this work is to explain this variance in disintegration slowdown. What causes the performance of some applications to be penalized so much more than others? More concretely, we search for application characteristics (x-axis values) that *are* correlated with this slowdown (y-axis) to explain the performance penalty of disintegration. We focus on the $P_d(A)/P_m(A)$ ratio, and defer to prior work for C_d/C_m [31, 41, 42, 43, 44].

3 EXPERIMENTAL METHODOLOGY

Modeled manycores: We modify the open-source[2] cycle-level, event-driven Swarm [22, 23] simulator based on Pin [30, 34] to evaluate the impact on performance of 256-core disintegrated systems relative to the monolith shown in Fig. 2. Parameters are given in Table 1. We use detailed core, cache, network, and main memory models, and do not use any Swarm task management or speculation as all benchmarks are multithreaded.

For disintegrated systems, we partition the monolithic 64-tile network into 16 chiplets of four tiles each, providing higher yield than four larger chiplets. Intel's Sapphire Rapids comprises four chips [32], Tenstorrent's Grendel forecasts eight [29], AMD's EPYC has nine [31] to thirteen [25], and NVIDIA's Simba has 36 [36].

The disintegrated interconnect topologies comprise two layers. The network-on-chip (NoC) uses a mesh on each die (4-node ring on `cmesh` and `mcm`), similar to products and prototypes from industry [5, 15, 18, 32, 36]. The network-on-package (NoP) topology is influenced by the disintegration technology. Fig. 5a and Fig. 5b show the combined NoP and NoC topologies for our active silicon interposer (`cmesh`) and multi-chip module (`mcm`) systems, respectively. We use a misaligned concentrated mesh for the interposer [24]. Concentration drives down active interposer area while misalignment improves average hop count and packet latency. Red intra-chip links connect black on-chip routers. Dashed blue links connect

[2] https://github.com/SwarmArch/sim

Table 1: Configuration of the 256-core system

Cores	256 cores in 64 tiles (4 cores/tile), 2 GHz, x86-64 ISA, 1 thread/core (like AmpereOne [2]);
Frontend	2-level bpred with 1024×10-bit BHSRs + 1024×2-bit PHT
Backend	2-way issue/rename/dispatch/commit, 36-entry issue buffer, 72-entry ROB, 16-entry ld/st queue (similar to KNL [40])
L1 caches	32 KB, per-core, split D/I, 8-way, 2-cycle latency
L2 caches	1 MB, per-tile, 8-way, inclusive, 9-cycle latency
LLC	256 MB, shared, static NUCA [26] (4 MB bank/tile), 16-way, inclusive, 12-cycle bank latency
Coherence	MESI, 64 B lines, in-cache directories
Main mem	16 controllers at chip edges, 120-cycle latency, 30 GB/s/controller
Net topology	8×8 mesh on `monolith`, 2×2 mesh per disintegrated chip
NoC and Interposer	16 B links, 4 physical networks, XY routing, 1 cycle/hop when going straight, 2 cycles on turns (like Tile64 [45]), 1 cycle per router
`mcm` NoP	1 cycle each way between chip and package substrate + 3 cycles for traversal of package + 1 cycle for clock domain crossing
`cmesh` NoP	1 cycle to cross between chip and interposer + 1 cycle for clock domain crossing (like Kite [7])

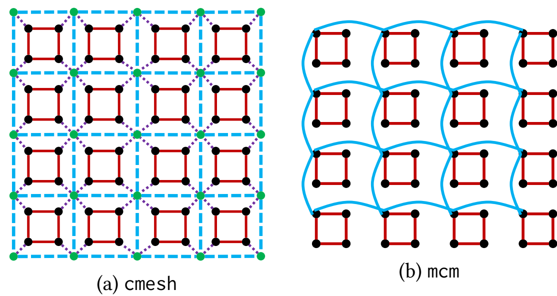


Figure 5: `cmesh` and `mcm` network topologies with a 4×4-chip, 2×2-tile/chip, 4-cores/tile configuration. `cmesh` uses a concentration factor of 4 and 5×5 XY-misalignment.

green on-interposer routers. Dotted purple links cross from on-chip to on-interposer routers. The `mcm` topology resembles the Simba network [36]. Blue links connect the chips through the package substrate. Each chip has one router designated for off-chip traffic; all off-chip-bound messages are routed here first.

We implement a hierarchical, X-first routing algorithm: starting from the source router, *(i)* route to the local off-chip router, *(ii)* route within the NoP to the off-chip router that is local to the destination, *(iii)* route to the destination. A packet skips any of these steps if already completed (i.e., exploits intra-chip routing when possible). A longer version of this work details `cmesh` and `mcm` routing [9].

We configure intra-chip and inter-chip router and link delays similarly to recent work. We add one cycle for clock-domain crossing when a packet moves to a different chip [7]. Link distances in the interposer and package substrate are approximately one side of a chip plus the distance between chips. Assuming this wire length is less than 10 mm for chips in 11 nm, and that the metal wire in the package substrate has similar length to one in a passive interposer, we use one cycle per link in the active interposer and three cycles per link in the package substrate [42].

Benchmarks: The shift to disintegration hampers the NoP. Consequently, we use a diverse set of 29 multithreaded benchmarks drawn from `PARSEC` [8] (`blackscholes`, `canneal`), `Splash-2x` [47] (`barnes`, `fft`, `lu_cb`, `lu_ncb`, `ocean_cp`, `ocean_ncp`, `radix`), `Splash-4` [19] (all except `fmm`), `PBBSv1` [39] (`mis`), `PBBSv2` [3] (`mm`, `msf`, `sf`, `sa`), and other graph benchmarks (`cf` [38], `color` [21]). These exhibit a diversity of parallel and data sharing patterns (Sec. 4). We use standard input sets for `PARSEC` and `Splash`, trigramString_1M as input to `sa`, movielens-1m [20] (10.0k vertices, 2.00M edges) as input to `cf`, and East USA roads [1] (3.60M vertices, 8.78M edges) and com-youtube [28] (1.16M vertices, 2.99M edges) as inputs to remaining benchmarks.

We exclude benchmarks from `Splash` and `PARSEC` that led to errors. For example, some excluded benchmarks have internal maximum thread counts that preclude 256-thread execution. We use the Cilk runtime for `cf`, `color`, and `mis`, and pthreads for all other benchmarks. Because our Pin-based simulator does not capture cycles in system calls, we replace the `pthread_mutex_t` in `Splash-2x` with a TTAS lock.

We fast-forward each benchmark to the start of its parallel region and simulate the full system for the entire region. We perform ten runs per input for all benchmarks on all three systems. Each point in a scatter plot represents one (benchmark,input) pair and is constructed from the average of run times and/or relevant metric.

4 RESULTS

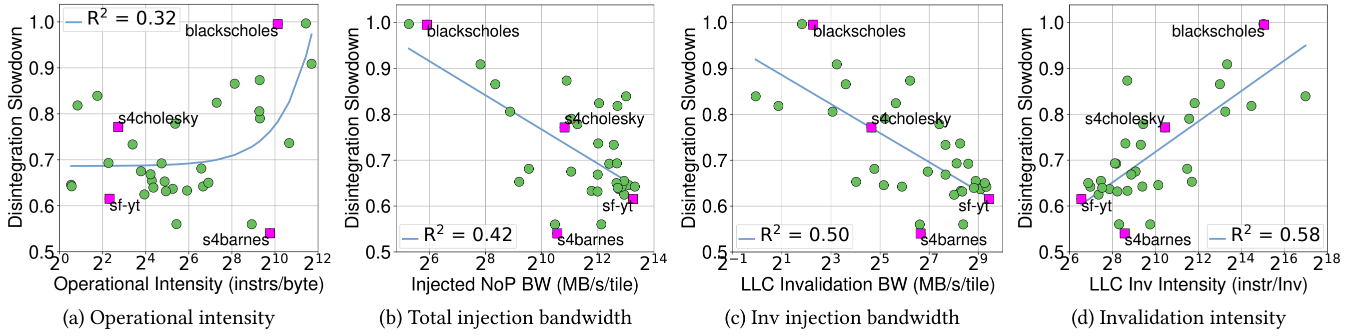
What architectural or application-level metrics explain the variance in disintegration slowdown? We consider two general categories: *(i)* classic performance metrics and *(ii)* network and data sharing metrics. As in Fig. 4, we plot our proposed metrics against the disintegration slowdown, perform linear regression, and measure the correlation. All x-axis metrics are measured on the monolithic system, as we find it typically gives better correlation; the monolith is less constrained, so it better satisfies the resources an application wants. We consider linear and exponential correlations and report the better of the two.

Table 2 summarizes our findings, reporting the correlations from each scatter plot. A longer version of this work analyzes every plot leading to a cell in this table [9]. For brevity, we showcase scatter plots for key metrics of interest in Fig. 6 and only for `mcm`.

4.1 Performance metrics poorly predict disintegration slowdown

Workload characterization often measures average instructions per cycle (IPC), consumed memory bandwidth, and operational intensity. However, we find all three poorly predict disintegration slowdown (i.e., correlation is low) [9]. These metrics are core- and memory-centric and are less sensitive to the NoP in particular.

We plot disintegration slowdown vs. operational intensity in Fig. 6a, as it gives the best R^2 of 0.32 on `mcm`. Deviating from Williams et al. [46], we compute operational intensity as the number of instructions per byte of data accessed from memory, as not all benchmarks use floating point. The line of best fit looks exponential on the logarithmic x-axis. There is a weak trend of reduced slowdown with increasing operational intensity, but this metric is insufficient to explain the variance in disintegration slowdown.



(a) Operational intensity (b) Total injection bandwidth (c) Inv injection bandwidth (d) Invalidation intensity

Figure 6: Comparing disintegration slowdown of `mcm` vs. key metrics measured on the monolith

Table 2: Application metrics and their correlation with disintegration slowdown

Metric Group	Metric	Correlation (`cmesh`)	Correlation (`mcm`)
Performance	IPC	0.26	0.14
	Consumed Memory Bandwidth	0.02	0.13
	Operational Intensity [46]	0.18	0.32
Base Network	Network Injection Bandwidth	0.31	0.42
	Average Network Latency	0.06	0.06
Network Injection Breakdown	GetS	0.32	0.41
	GetX	0.40	0.49
	Inv (Invalidate)	**0.48**	0.50
	InvX (Downgrade)	0.46	**0.57**
	Data Response	0.34	0.46
	PutS (Clean Eviction)	0.00	0.10
	PutX (Dirty Eviction)	0.04	0.05
Data Sharing	# Read Sharers	0.04	0.07
	Invalidation Intensity	**0.63**	0.58
	Downgrade Intensity	0.55	**0.59**
	Sharing Fraction [16]	0.34	0.31

4.2 Network and data sharing metrics are better

Since the NoP is the key system component constrained by disintegration, we consider the correlation of disintegration slowdown with average network injection bandwidth from tiles and average packet latency. Unsurprisingly, Fig. 6b shows that benchmarks better utilizing the monolith network (injecting more) have more performance degradation with disintegration, however, the correlation is weak. Table 2 reports almost no correlation with latency.

We break down network traffic by components and find stronger correlation of disintegration slowdown with invalidation (Inv) and downgrade (InvX) traffic. Fig. 6c plots disintegration slowdown vs. the bandwidth of Inv messages injected from the LLC. Inv and InvX requests are caused by read-write sharing and synchronization among threads. Ideally, such data would be perfectly reused (shared) by threads before being evicted off chip, reducing off-chip memory traffic. Since these messages traverse the NoP, their bandwidth should correlate with performance impact, given the slower disintegrated network. Evidently, with $R^2 \approx 0.5$, this does not tell the entire story.

We develop *invalidation intensity* as a sharing analogue for operational intensity: instructions per Inv message, rather than instructions per byte from memory. This metric is a proxy for the period, in instructions, at which readers of data are invalidated due to sharing. Fig. 6d plots disintegration slowdown vs. invalidation intensity. Since frequency is constant in our model, Fig. 6d ultimately factors out average IPC from Fig. 6c. With an invalidation rate normalized by instructions rather than cycles, invalidation intensity is less (albeit still) dependent on microarchitecture. Fig. 6d shows that disintegration slowdown worsens as the invalidation intensity increases: more sharing causes more performance penalty.

5 CONCLUSION

In the pursuit of larger and higher performance systems, the microprocessor industry is shifting to disintegration: reassembling smaller chiplets within a package. This improves yield and reduces cost, with one key goal of improving performance per dollar for customers. However, the performance penalty due to running applications on disintegrated systems has not been well studied. This paper presented a characterization of the impacts of disintegration on application performance, focusing on active silicon interposer and package substrate (`mcm`) interconnects. On average, application scaling remains similar between the monolith and disintegrated designs, although the cheaper the disintegration technology, the worse the average performance penalty. However, the per-application normalized disintegration slowdown has no correlation with monolithic performance and has high variance, with some applications experiencing nearly twice the average performance slowdown. We introduced *invalidation intensity* as a metric for data sharing rate and find it correlates with disintegration slowdown with R^2 of about 0.6, stronger than correlation with injected network bandwidth. These results corroborate our intuition that data sharing, in particular, stresses the network because accesses to shared data are not further limited by memory bandwidth. Future work could continue the search for a near-perfect predictor of disintegration slowdown, ideally using microarchitecture-independent metrics, then leverage those insights to develop architectural and software mechanisms to mitigate disintegration slowdown.

6 ACKNOWLEDGMENTS

We sincerely thank Tarek Abdelrahman, Javad Abdi, Natalie Enright Jerger, Gilead Posluns, Guowei Zhang, Guozheng Zhang, and the anonymous reviewers for their helpful feedback. Some text in this paper was edited using ChatGPT. This work was supported in part by the Digital Research Alliance of Canada, the University of Toronto, NSERC, and an Ontario QEII-GSST.

REFERENCES

[1] 2006. *9th DIMACS Implementation Challenge: Shortest Paths*. http://www.dis.uniroma1.it/~challenge9, archived at https://perma.cc/5KYT-YM36. (2006).

[2] The Ampere Team. 2023. *Ampere Computing unveils new AmpereOne processor family with 192 custom cores*. Ampere Computing Press. (May 2023).

[3] Daniel Anderson, Guy E. Blelloch, Laxman Dhulipala, Magdalen Dobson, and Yihan Sun. 2022. "The problem-based benchmark suite (PBBS), v2." In *PPoPP*.

[4] Apple. 2022. *Apple unveils M1 Ultra, the world's most powerful chip for a personal computer*. Apple Newsroom. (2022).

[5] Mohamed Arafa, Bahaa Fahim, Sailesh Kottapalli, Akhilesh Kumar, Lily P. Looi, Sreenivas Mandava, Andy Rudoff, Ian M. Steiner, Bob Valentine, Geetha Vedaraman, and Sujal Vora. 2019. "Cascade Lake: next generation Intel Xeon scalable processor." *IEEE Micro*, 39, 2.

[6] Akhil Arunkumar, Evgeny Bolotin, Benjamin Cho, Ugljesa Milic, Eiman Ebrahimi, Oreste Villa, Aamer Jaleel, Carole-Jean Wu, and David Nellans. 2017. "MCM-GPU: multi-chip-module gpus for continued performance scalability." In *ISCA-44*.

[7] Srikant Bharadwaj, Jieming Yin, Bradford Beckmann, and Tushar Krishna. 2020. "Kite: a family of heterogeneous interposer topologies enabled via accurate interconnect modeling." In *DAC-57*.

[8] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. "The PARSEC benchmark suite: characterization and architectural implications." In *PACT-17*.

[9] Isidor R. Brkić. 2023. *The Performance Cost of Disintegrated Manycores: Which Applications Lose and Why?* Master's thesis. University of Toronto.

[10] Grigory Chirkov and David Wentzlaff. 2023. "Seizing the bandwidth scaling of on-package interconnect in a post-Moore's law world." In *ICS'23*.

[11] Mark Cianchetti, Nicolás Sherwood-Droz, and Christopher Batten. 2010. "Implementing system-in-package with nanophotonic interconnect." In *WINDS*.

[12] Ayse Coskun, Furkan Eris, Ajay Joshi, Andrew B. Kahng, Yenai Ma, and Vaishnav Srinivas. 2018. "A cross-layer methodology for design and optimization of networks in 2.5d systems." In *ICCAD* Article 101.

[13] J.A. Cunningham. 1990. "The use and evaluation of yield models in integrated circuit manufacturing." *IEEE TSM*, 3, 2.

[14] Debendra Das Sharma, Gerald Pasdast, Zhiguo Qian, and Kemal Aygun. 2022. "Universal chiplet interconnect express (UCIe): an open industry standard for innovations with chiplets at package level." *IEEE TCPMT*, 12, 9.

[15] Mark Evers, Leslie Barnes, and Mike Clark. 2022. "The AMD next-generation "Zen 3" core." *IEEE Micro*, 42, 3.

[16] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A.D. Popescu, A. Ailamaki, and B. Falsafi. 2012. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." In *ASPLOS-XVII*.

[17] Brian Gaide, Dinesh Gaitonde, Chirag Ravishankar, and Trevor Bauer. 2019. "Xilinx adaptive compute acceleration platform: Versal architecture." In *FPGA*.

[18] Ofer Geva, Chris Berry, Robert Sonnelitter, David Wolpert, Adam Collura, Thomas Strach, Di Phan, Cedric Lichtenau, Alper Buyuktosunoglu, Hubert Harrer, Jeffrey Zitz, Chad Marquart, Douglas Malone, Tobias Webel, Adam Jatkowski, John Isakson, Dina Hamid, Mark Cichanowski, Michael Romain, Faisal Hasan, Kevin Williams, Jesse Surprise, Chris Cavitt, and Mark Cohen. 2022. "IBM Telum: a 16-Core 5+ GHz DCM." In *ISSCC*.

[19] Eduardo José Gómez-Hernández, Juan M. Cebrian, Stefanos Kaxiras, and Alberto Ros. 2022. "Splash-4: a modern benchmark suite with lock-free constructs." In *IISWC*.

[20] F. Maxwell Harper and Joseph A. Konstan. 2015. "The MovieLens datasets: history and context." *ACM TiiS*, 5, 4.

[21] William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2014. "Ordering heuristics for parallel graph coloring." In *SPAA*.

[22] Mark C. Jeffrey, Suvinay Subramanian, Maleen Abeydeera, Joel Emer, and Daniel Sanchez. 2016. "Data-centric execution of speculative parallel programs." In *MICRO-49*.

[23] Mark C. Jeffrey, Suvinay Subramanian, Cong Yan, Joel Emer, and Daniel Sanchez. 2015. "A scalable architecture for ordered parallelism." In *MICRO-48*.

[24] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H. Loh. 2015. "Enabling interposer-based disintegration of multi-core processors." In *MICRO-48*.

[25] Chris Karamatas. 2023. *AMD EPYC 9004 processor architecture overview*. AMD Documentation Hub. (June 2023).

[26] Changkyu Kim, Doug Burger, and Stephen W. Keckler. 2002. "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches." In *ASPLOS-X*.

[27] Gary Lauterbach. 2021. "The path to successful wafer-scale integration: the Cerebras story." *IEEE Micro*, 41, 6.

[28] Jure Leskovec and Andrej Krevl. 2014. *SNAP datasets: Stanford large network dataset collection*. http://snap.stanford.edu/data. (2014).

[29] Wei-han Lien. 2023. "From mW to MW: scalable RISC-V processors for AI everywhere." In *Proc. of the RISC-V Summit Europe*. (May 2023).

[30] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. 2005. "Pin: Building customized program analysis tools with dynamic instrumentation." In *PLDI*.

[31] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. 2021. "Pioneering chiplet technology and design for the AMD EPYC and Ryzen processor families : industrial product." In *ISCA-48*.

[32] Nevine Nassif, Ashley O. Munch, Carleton L. Molnar, Gerald Pasdast, Sitaraman V. Lyer, Zibing Yang, Oscar Mendoza, Mark Huddart, Srikrishnan Venkataraman, Sireesha Kandula, Rafi Marom, Alexandra M. Kern, Bill Bowhill, David R. Mulvihill, Srikanth Nimmagadda, Varma Kalidindi, Jonathan Krause, Mohammad M. Haq, Roopali Sharma, and Kevin Duda. 2022. "Sapphire Rapids: the next-generation Intel Xeon scalable processor." In *ISSCC*.

[33] Kunle Olukotun and Lance Hammond. 2005. "The future of microprocessors." *ACM Queue*, 3, 7.

[34] Heidi Pan, Krste Asanović, Robert Cohn, and Chi-Keung Luk. 2005. "Controlling program execution through binary instrumentation." *SIGARCH Computer Architecture News*, 33, 5.

[35] Hesam Shabani and Xiaochen Guo. 2019. "ClusCross: a new topology for silicon interposer-based network-on-chip." In *NOCS*.

[36] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019. "Simba: scaling deep-learning inference with multi-chip-module-based architecture." In *MICRO-52*.

[37] Scott Shenker. 1995. "Fundamental design issues for the future internet." *IEEE JSAC*, 13, 7.

[38] Julian Shun and Guy E Blelloch. 2013. "Ligra: a lightweight graph processing framework for shared memory." In *PPoPP*.

[39] Julian Shun, Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Aapo Kyrola, Harsha Vardhan Simhadri, and Kanat Tangwongsan. 2012. "Brief announcement: the problem based benchmark suite." In *SPAA*.

[40] Avinash Sodani, Roger Gramunt, Jesus Corbal, Ho-Seop Kim, Krishna Vinod, Sundaram Chinthamani, Steven Hutsell, Rajat Agarwal, and Yen-Chen Liu. 2016. "Knights Landing: Second-generation Intel Xeon Phi product." *IEEE Micro*, 36, 2.

[41] Dylan Stow, Itir Akgun, Russell Barnes, Peng Gu, and Yuan Xie. 2016. "Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5d/3d integration." In *ICCAD*.

[42] Dylan Stow, Itir Akgun, and Yuan Xie. 2019. "Investigation of cost-optimal network-on-chip for passive and active interposer systems." In *SLIP*.

[43] Dylan Stow, Yuan Xie, Taniya Siddiqua, and Gabriel H. Loh. 2017. "Cost-effective design of scalable high-performance systems using active and passive interposers." In *ICCAD*.

[44] Lisa Su and Sam Naffziger. 2023. "Innovation for the next decade of compute efficiency." In *ISSCC*.

[45] David Wentzlaff, Patrick Griffin, Henry Hoffmann, Liewei Bao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chyi-Chang Miao, John F. Brown III, and Anant Agarwal. 2007. "On-chip interconnection architecture of the Tile Processor." *IEEE Micro*, 27, 5.

[46] Samuel Williams, Andrew Waterman, and David Patterson. 2009. "Roofline: an insightful visual performance model for multicore architectures." *Commun. ACM*, 52, 4.

[47] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. "The SPLASH-2 programs: characterization and methodological considerations." In *ISCA-22*.

[48] Florian Zaruba, Fabian Schuiki, and Luca Benini. 2021. "Manticore: a 4096-core RISC-V chiplet architecture for ultraefficient floating-point computing." *IEEE Micro*, 41, 2.

[49] Shiqing Zhang, Mahmood Naderan-Tahan, Magnus Jahre, and Lieven Eeckhout. 2023. "SAC: sharing-aware caching in multi-chip GPUs." In *ISCA-50* Article 43.

[50] Hao Zheng, Ke Wang, and Ahmed Louri. 2020. "A versatile and flexible chiplet-based system design for heterogeneous manycore architectures." In *DAC-57*.