Finding Balance in the post-Moore's Law Era

Jeffrey Young, *Member, IEEE,* and Rich Vuduc, *Member, IEEE*

Abstract—Recent developments in 3D stacked memory technologies and the promise of fast optical interconnects have offered new possibilities for balancing communication and computation time in future supercomputers. However, the end of Moore's Law means that future architectural improvements will become more limited in scope. We theoretically evaluate a proposal that balanced "post-Moore's" systems will be most achievable via a combination of small, lightweight processors, high-bandwidth memories with optimized "horizontal" communication, and network-oriented algorithm codesign.

1 Introduction

As we move towards exascale computing and beyond, technology limitations of the post-Moore's Law era mean that we will need to allocate a finite supply of transistors and energy by creating both balanced and scalable supercomputers. In previous work, we postulated that a lightweight, CPU-style system would achieve the best balance (ie, flop to byte ratio) but also that newer xPU stacked memories and network topologies might change this analysis [3], [4]. Emerging technologies, such as stacked memories like High Bandwidth Memory (HBM) and Hybrid Memory Cube (HMC), and first-generation optical networks, such as NVLink and Omni-Path, provide us with new architectural possibilities for designing future systems with a better balance of computation and fast data movement.

In part due to these architectural trends, we argue that *scalable, balanced supercomputers will have to focus more on memory-centric architectures and emerging network designs, while algorithms will need to be optimized in tandem with horizontal or network-based movement*. In addition, we speculate that as the slowdown in Moore's Law reduces the benefit of increased cache sizes and the possibility of higher peak flop rates, maximizing local and network bandwidth becomes the most important factor for creating a balanced system.

A recent position paper suggests that although stacked memories will improve memory bandwidth, future applications still might not have enough memory-level parallelism (MLP) to take advantage of larger amounts of local bandwidth [10]. In addition, design-focused proposals argue that future systems using stacked memories with silicon interposers need to be designed more like a Network on Chip (NoC) in order to mitigate thermal concerns while also providing appropriate amounts of local bandwidth and storage [9]. Both of these memory-related papers point to two concerns that we believe are important for balanced system design in the post-Moore's era: 1) balanced systems need to be aggressively networked with high-bandwidth memories to provide flexibility in maximizing MLP, and 2) algorithms need to be designed cooperatively with local and global networks in order to best balance computation and communication.

As part of our argument, we address trends in future systems by looking closely at the costs for an algorithm running on an abstract distributed memory system. Distributed algorithms face two communication costs: vertical communication through the memory hierarchy and horizontal communication across the network [11]. We suggest that, as the problem size grows, vertical communication becomes a fundamental bottleneck whereas exploiting distributed memory can actually lower asymptotic horizontal communication costs. Additionally, balanced processors having a low flop-to-byte ratio can help keep vertical communication costs in check. These observations are the critical motivators for the kind of stacked lightweight system we advocate.

- *School of Computer Science, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: jyoung9@gatech.edu*
- *School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, 30332.E-mail: richie@cc.gatech.edu*

PMES Workshop, Salt Lake City, 14 Nov 2016. `http://j.mp/pmes2016`

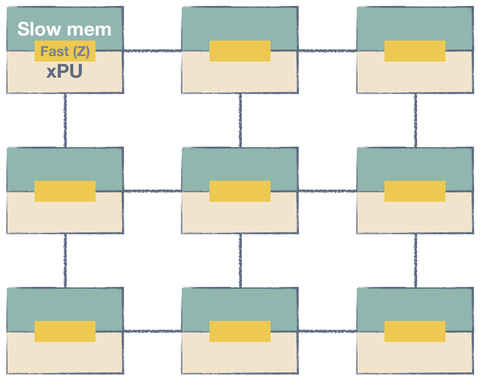

Fig. 1. An abstract distributed memory machine where each node has a two-level memory hierarchy.

2 Scalable Systems are Horizontal and Balanced

To make our argument for lightweight systems with enhanced horizontal communication more precise, we generalize an earlier, primarily theoretical analysis of the 3-D fast Fourier transforms (FFTs), which are a communication-intensive proxy for data-intensive algorithms more broadly, like sorting [3], [12]. Consider the model of a distributed memory machine shown in Figure 1 and refer to the symbols in Table 1. This machine has P compute nodes connected by some network, say, a d-dimensional mesh or torus (a $d = 2$ mesh is shown in the figure). Each node is a processor with a two-level memory hierarchy consisting of a slow memory and a fast memory, where the size of the fast memory is Z.

Further suppose that this machine has the following speeds and feeds. Let R_0 denote the processor's peak performance in operations per unit time, which is achievable when the data resides in fast memory; thus, the machine's peak performance is $R_{\max} = R_0 P$. Let B_{mem} denote the per-processor memory bandwidth in words per unit time; thus, the machine's aggregate main memory bandwidth across all nodes is $B_{\text{agg}} = B_{\text{mem}} P$. Lastly, let B_{link} denote the peak bandwidth

Symbol	Description
P	Number of compute nodes
d	Dimension of the toroidal network
Z	Size of a node's "fast" memory [words]
R_0	Peak performance per processor [flops / time]
B_{mem}	Main memory bandwidth [words / time]
B_{link}	Link bandwidth [words / time]
$R_{\max}$	Machine peak [flops / time]
B_{agg}	Aggregate memory bandwidth [words / time]
B_{bisect}	Bisection bandwidth [words / time]
F	Total # of floating-point operations [flops]
Q_{mem}	Total slow-fast memory transfers [words]
Q_{net}	Total # of network transfers [words]
$g(Z)$, $h_1(P,d)$, $h_2(n)$	Algorithm-specific, non-decreasing functions, e.g., $\sqrt{Z}$, $P^{\frac{d-1}{d}}$, $\log n$.

TABLE 1
Symbols

per network link, also in words per unit time; thus, for a d-dimensional torus, the bisection bandwidth of the machine will be approximately $B_{\text{bisect}} \approx \mathcal{O}\left(P^{\frac{d-1}{d}} \cdot B_{\text{link}}\right)$ [5].

To analyze an algorithm running on this machine, suppose it executes a total of F flops (aggregated across all nodes), moves a total of Q_{mem} words between slow and fast memory (vertical communication volume), and moves a total of Q_{net} words between nodes (horizontal communication volume). Note that these counts will generally be functions of the problem size, n, and the machine parameters Z, P, and d. Typically, they take the form of

$$F = F(n), \tag{1}$$

$$Q_{\text{mem}} = Q_{\text{mem}}(n; Z) = \frac{F(n)}{g(Z)}, \text{ and} \tag{2}$$

$$Q_{\text{net}} = Q_{\text{net}}(n; P) = F(n) \cdot \frac{h_1(P,d)}{h_2(n)}, \tag{3}$$

where g, h_1, and h_2 are all non-decreasing functions of their arguments. In particular, $g(Z)$ models the potential of increasing fast memory capacity to decrease Q_{mem}; $h_1(P,d)$ models potential increases in network communication volume as the number of nodes increases;[1] and $F(n)/h_2(n)$ says that network communication volume can be asymptotically smaller than flops. For instance, a conventional dense matrix multiply has $F = \mathcal{O}(n^3)$; $g(Z) = \Omega\left(\sqrt{Z}\right)$ so that $Q_{\text{mem}} = \mathcal{O}\left(n^3/\sqrt{Z}\right)$ [7]; and $h_1(P,d) = \sqrt{P}$ (when $d = 2$) and $h_2(n) = \Omega(n)$ so that $Q_{\text{net}} = \mathcal{O}\left(n^2\sqrt{P}\right)$ [6]. By contrast, a 3-D FFT has $F = \mathcal{O}(n^3 \log n)$; $g(Z) = \Omega(\log Z)$ so that $Q_{\text{mem}} = \mathcal{O}(n^3 \log n / \log Z)$ [1]; and $h_1(P,d) = \mathcal{O}(1)$ (when $d \geq 3$) and $h_2(n) = \Omega(\log n)$ so that $Q_{\text{net}} = \Omega(n^3)$ [3].[2]

These functions have two critical properties. First, $g(Z)$ is typically a sublinear function of Z; as such, continually increasing fast memory capacity will show severely diminishing returns. Secondly, observe how these costs behave with F: while Q_{mem} might decrease with increasing Z, it is usually asymptotically identical to F; and while Q_{net} might increase with increasing P, it can also be asymptotically lower than F.

What do these facts imply? Consider an estimate of the running time of the algorithm. Assuming perfect parallelism, the minimum time to execute the flops is $T_{\text{comp}} = F/R_{\max}$, the time for the memory references is $T_{\text{mem}} = Q_{\text{mem}}/B_{\text{agg}}$, and the time for all network communication is approximately $T_{\text{net}} \approx Q_{\text{net}}/B_{\text{bisect}}$. After some algebra, the communication times relative to the compute time are

$$\frac{T_{\text{mem}}}{T_{\text{comp}}} = \frac{R_0}{B_{\text{mem}}} \times \frac{1}{g(Z)} \tag{4}$$

$$\text{and } \frac{T_{\text{net}}}{T_{\text{comp}}} \approx \frac{R_0}{B_{\text{link}}} \times \frac{P^{\frac{1}{d}} \cdot h_1(P,d)}{h_2(n)}. \tag{5}$$

Equations (4) and (5) summarize the tradeoffs affecting communication costs. Regarding the memory hierarchy, decreasing the flop-to-byte ratio (R_0/B_{mem}) or increasing the fast memory size help reduce cost; however, recall $g(Z)$ tends to grow slowly with Z. Regarding the network, decreasing the (analogous) flop-to-byte ratio of R_0/B_{link} also helps. And while network cost may grow with increasing P via $h_1(P,d)$, that cost should be weighed against the potential asymptotic reductions that come from a distributed algorithm via $h_2(n)$.

One caveat is that if the network energy is substantially higher than local memory energy, then distributing more aggressively—to *increase* horizontal communication while *reducing* vertical communication—may actually yield less energy-efficient systems [3]. This tradeoff will likely be heavily influenced by the success or failure of optical interconnects, which could make horizontal communication more competitive in terms of network energy usage while also providing better system balance.

3 Finding Balance As Moore's Law Concludes

Looking forward at long-term trends in memory-centric system and algorithm design, we have proposed two new post-Moore "iron laws," equations (4) and (5). In the most extreme post-Moore's scenario, technology scaling stops, and Equation (4) becomes either slow-growing or constant. This change occurs because Z cannot increase since there are no transistors left to increase the size of fast memory and because R_0 plateaus due to the lack of transistor scaling. For this specific case, increasing B_{mem} and modifying the algorithm-specific knob of $g(\cdot)$ via locality-aware algorithm design become the primary methods to improve system balance, at least until B_{mem} also plateaus, likely one to two generations after R_0. In the network case outlined in Equation (4), designing a balanced system will require coordinated design choices as to the dimensionality of the network, P and $h_1(\cdot)$, and algorithmic optimization of $h_2(\cdot)$. In summary, system balance can be improved by limited scaling to B_{mem} and B_{link} even after technology scaling stops for processor technology, and further algorithmic optimization provides the best avenue for improvement when architectural modifications become too costly.

Previous approaches to system design have captured some of these design characteristics, specifically the BlueGene [2], [8] series of machines that focused on maximizing the ratio of compute nodes to B_{link} through the use a 5-D torus ($d = 5$) and routing logic integrated onto a System on Chip. However, recent trends in supercomputer design have at least temporarily moved back towards a combination of lower-dimension, commodity interconnects like PCI Express, InfiniBand, and NVLink, which may end up reducing system balance with respect to powerful new GPU and Phi accelerators.

While these two equations cannot capture the full scope of system design in the post-Moore's era, they do indicate that our research focus should be on optimizing local and global interconnects and that *coordinated* algorithmic changes are required in order to make the best use of limited resources in future supercomputers.

1. For example, a broadcast or reduction grows like $\mathcal{O}(\log P)$.
2. This analysis assumes the so-called "transpose" algorithm.

REFERENCES

[1] A. Aggarwal and S. Vitter, Jeffrey, "The input/output complexity of sorting and related problems," *Communications of the ACM*, vol. 31, no. 9, pp. 1116–1127, aug 1988. [Online]. Available: http://portal.acm.org/citation.cfm?doid=48529.48535

[2] D. Chen, N. Eisley *et al.*, "Looking under the hood of the ibm blue gene/q network," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 69:1–69:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389090

[3] K. Czechowski, C. McClanahan *et al.*, "On the communication complexity of 3D FFTs and its implications for exascale," in *Proc. ACM Int'l. Conf. Supercomputing (ICS)*, San Servolo Island, Venice, Italy, June 2012.

[4] K. Czechowski and R. Vuduc, "A theoretical framework for algorithm-architecture co-design," in *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013.

[5] A. Grama, A. Gupta *et al.*, *Introduction to Parallel Computing*. Addison-Wesley, 2003.

[6] D. Irony, S. Toledo, and A. Tiskin, "Communication lower bounds for distributed-memory matrix multiplication," *Journal of Parallel and Distributed Computing*, vol. 64, no. 9, pp. 1017–1026, sep 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0743731504000437

[7] H. Jia-Wei and H. T. Kung, "I/O complexity: The red-blue pebble game," in *Proceedings of the thirteenth annual ACM symposium on Theory of computing - STOC '81*. New York, New York, USA: ACM Press, may 1981, pp. 326–333. [Online]. Available: http://portal.acm.org/citation.cfm?doid=800076.802486

[8] D. J. Kerbyson, K. J. Barker *et al.*, "Tracking the performance evolution of Blue Gene systems," in *Proceedings of 28th International Supercomputing Conference*, 2013, pp. 317–329. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38750-0_24

[9] G. H. Loh, N. E. Jerger *et al.*, "Interconnect-memory challenges for multi-chip, silicon interposer systems," in *Proceedings of the 2015 International Symposium on Memory Systems*, ser. MEMSYS '15. New York, NY, USA: ACM, 2015, pp. 3–10. [Online]. Available: http://doi.acm.org/10.1145/2818950.2818951

[10] M. Radulovic, D. Zivanovic *et al.*, "Another trip to the wall: How much will stacked DRAM benefit HPC?" in *Proceedings of the 2015 International Symposium on Memory Systems*, ser. MEMSYS '15. New York, NY, USA: ACM, 2015, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/2818950.2818955

[11] E. Solomonik, "Provably efficient algorithms for numerical tensor algebra," Ph.D. dissertation, August 2014. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-170.pdf

[12] R. Vuduc and K. Czechowski, "What GPU computing means for high-end systems," *IEEE Micro*, vol. 31, no. 4, pp. 74–78, July/August 2011.