Impact of Computing-in-Memory on the Performance of Processor-and-Memory Hierarchies*

Renato J. O. Figueiredo†, José A. B. Fortes†, Zina Ben Miled‡,
Valerie Taylor†† and Rudolf Eigenmann†

†School of Electrical and Computer Engineering
1285 Electrical Engineering Building
Purdue University
West Lafayette, IN 47907-1285

‡Department of Electrical Engineering
Purdue University
Indianapolis, IN 46202

††Department of EECS
Northwestern University
Evanston, IL 60208-3118

[0]This research was supported in part by NSF grants ASC-9612133 and ASC-9612023. Renato Figueiredo is also supported by a CAPES grant.

Abstract

A Hierarchy of Processor-And-Memory can be viewed as an extension of the notion of a memory hierarchy. The extension entails the inclusion of processors with different performance in different levels of the memory hierarchy. Technology trends, applications behavior and previous research suggest that a multiprocessor system organized as a Hierarchy of Processor-And-Memory (HPAM) may offer considerable advantages over conventional multiprocessor organizations. This paper analyzes the performance impact of computing-in-memory, for a multi-level HPAM system, as compared to a conventional multiprocessor system with the same cost. The analysis entails using performance models and simulation data. The performance models capture the important differences in the data access time between the two approaches. Underlying the comparative study is the assumption that the cost of a processor is in proportion to the square of its performance, also known as Grosch's law. The 9 benchmarks used in the evaluations belong to the CMU, Perfect Club and SPEC95 suites. While the evaluation methodology takes into account the heterogeneity of HPAM, the emphasis is placed on modeling the impact of computing-in-memory on the relative performance of the multiprocessors under study. The results indicate that, with rare exceptions, HPAM outperforms conventional multiprocessor designs (of identical cost) by as much a 80% in the benchmarks and the ranges of model parameters considered in the study. In the very few exceptions to this conclusion HPAM is never outperformed by more than 20%.

Contents

List of Figures

List of Tables

1 Introduction

A system organized as a Hierarchy of Processor-And-Memory (HPAM) [35, 32, 33] can be viewed as an extension of the familiar notion of memory hierarchy. The extension consists of including processors into one or more levels of the memory hierarchy. Assuming that the top (i.e. first) level of the hierarchy is the fastest, any given memory level is extended with processors that are slower, less expensive and in larger number than those in the preceding level. Heterogeneity and computing-in-memory are thus inherent to an HPAM system. This paper analyzes and reports on the performance of simulated HPAM machines against "flat" (i.e. homogeneous) multiprocessor systems.

There is considerable evidence that differences in the rate of improvement of processor, memory and I/O technology may call for novel computer organizations within the next 10 years. In particular, the increasing gap between CPU cycle times and memory access latency has motivated several proposals for either "bringing the processor to the (DRAM) memory" [5, 24, 23, 19] or "bringing the memory (SRAM) to the processor" [3]. The main purpose of the proposed designs is to avoid the so-called "memory wall" [30, 7, 18] as well as the "memory bandwidth wall" [20].

An HPAM also relies on computing-in-memory in order to avoid long memory access latencies. The analogy to memory hierarchy is present in two forms. One, already mentioned in the first paragraph, is the use of larger numbers of lower cost, slower processors in lower levels of the hierarchy (just as larger, less expensive slower memory technology is found in lower levels of the memory hierarchy). The second analogy is in relation to locality of references, the fundamental property of real programs that well-designed memory hierarchies exploit in order to provide small average access times. In [32] and subsequent studies (including this paper) it has been empirically established that there exists spatial and temporal locality of (degree of) parallelism. This behavior is exploited by HPAM by keeping "parallel" data "close" to the processors in the HPAM level that can process it with the desired degree of parallelism and avoiding unnecessary data movement between levels.

In a conventional multiprocessor system all processors are identical. Slight exceptions include the IBM SP2 [8] with its thin and thick nodes and the T3D [2] which has a single front-end processor (Cray C-90 or Y-MP) and a back-end homogeneous multiprocessor (with Alpha processors). According to [17] , as long as the cost and performance of processors (and implicitly the associated memory system) are linearly related, the best multiprocessor system is one which consists of as many of the best available processors as allowed by a given system budget.

However, also according to [17], a superlinear relation (e.g. quadratic as in Grosch's law [10, 11, 22]) between cost and performance renders homogeneous solutions non-optimal for large enough number of processors. There is an optimal point in the cost-performance function (of the number of processors) beyond which homogeneous solutions are no longer desirable [12, 17, 4, 9]. The intuition behind this conclusion is related to Amdahl's law. An heterogeneous machine executes sequential code on the fastest available processor and parallel code on the largest possible number of processors so that neither code segments become a bottleneck. In contrast, for a similar budget and a superlinear cost-to-performance relationship, an homogeneous machine would either use slow(er) processors for the sequential code segments or few(er) processors for the parallel code segments.

The relation between microprocessor cost and performance is a complex one which manufacturers consider to be proprietary information rarely to be disclosed. However, several arguments suggest that cost grows superlinearly with performance. The studies reported in [10, 11, 22] establish that cost is proportional to the square of performance for microprocessors in the same family. Assuming a (conservative) linear relationship between VLSI *area* and performance, yield-based cost models [13, 14] would suggest that chip cost increases in proportion to performance raised to α, a typical value being α=4. It is also predicted that future improvements in performance will incur dramatically high development costs (for both microprocessor design and fabrication processes) [26, 20]. Superlinear costs and the implication that homogeneous solutions may be suboptimal motivates the consideration of heterogeneity in future multiprocessor systems. Several studies [32, 35, 4, 17] have analyzed the potential cost-performance gains of

such systems assuming (conservatively) that Grosch's law holds (i.e. α=2). This assumption is also made for the HPAM design/models used in this paper.

The rationale and initial quantitative arguments for the HPAM concept appeared first in [32]. The novelty and complexity of HPAM has precluded the full simulation of such a system in order to study how heterogeneity and computing-in-memory contribute to HPAM's advantage over a conventional system. However, in [33], an execution-driven HPAM simulator (HPAM_Sim [34]) that simulates multiple processors and networks of different speeds was used to analyze the impact of heterogeneity on the performance of HPAM. In contrast, the purpose of this paper is to quantify and analyze the advantages of computing-in-memory (i.e. in different levels of HPAM) against computing on a conventional "flat" multiprocessor system. Performance data was gathered by using an execution-driven simulator that models memory behavior "in the large" of selected benchmarks from several suites (Spec95 [28], Perfect Club [15] and CMU [21]). Due to resource, time and complexity constraints, the simulator does not model in detail data movements inside an HPAM level. These effects are included indirectly as explained in Section 6.

The rest of this paper is organized as follows. Section 2 introduces the HPAM machine model. The execution time model of HPAM and flat machines is presented in Section 3. Section 4 presents the simulation framework and qualitative evaluation methodology. Section 5 presents the simulation results and performance analysis. Section 6 discusses the limitations of the performance model, and Section 7 presents conclusions of the paper.

2 HPAM Machine Model

A hierarchical processor-and-memory (HPAM) machine is a heterogeneous, multilevel parallel computer. Each HPAM level contains processors, memory and interconnection network. The speed and number of processors, latency and capacity of memories and network differ between levels in the hierarchy. The following characteristics hold across the different HPAM levels from top to bottom: individual processor performance decreases, number of processors increases, and memory/network latency and capacity increase.

Tasks are assigned to HPAM levels according to their *degree of parallelism* (DoP). Highly parallel code fractions of an application are assigned to bottom levels, where large number of processors and large memory capacity are available, while sequential and modestly parallel fractions are assigned to top levels, where a small number of fast processors and memories are available.

The HPAM approach to computing-in-memory bears similarities with IRAM and PIM efforts [5, 24], but it relies on heterogeneity and locality with respect to degree of parallelism to build a hierarchy of processor-and-memory subsystems where each subsystem is designed to be cost-efficient in its parallelism range. An MPP system implemented with dense, relatively inexpensive and slow memory technology, can be very efficient for highly parallel code, while a tightly-coupled symmetric multiprocessor containing a smaller amount of fast, expensive memory, can be very efficient for code mostly sequential or with moderate parallelism. The merging of these systems under the HPAM concept provides a cost-efficient solution for applications with different levels of parallelism.

In this paper, an HPAM machine with two processor-and-memory levels is studied, and its performance is compared to a homogeneous (also referred as "flat" throughout the paper) machine with the same estimated cost. Data obtained from the simulation of parallelized applications and an analytical execution time model are used to estimate the performance of both architectures.

The HPAM machine is specified only to the extent necessary to evaluate the impact of computing-in-memory. The relatively straightforward design is shown in Figure 1, along with the flat machine model. The homogeneous machine is composed of a number of interconnected processors with on-chip caches on the first level; extra (off-chip) cache storage on the second level; and main memory on the third level. In the HPAM machine, memory in the second level is augmented with processors. The second HPAM level is hence composed of connected processors with on-chip caches. The first level of HPAM contains only one processor, as powerful as one of the processors in the homogeneous configuration, and the second level contains several identical processors, less powerful than the processor in the first level (by about a factor of

two in the studies of this paper). Such configuration is referred as a *2-level HPAM* throughout this paper, since only two HPAM levels are "active" (i.e., contain processors). It is assumed, for simplicity, that the network connecting the different levels is a bus.

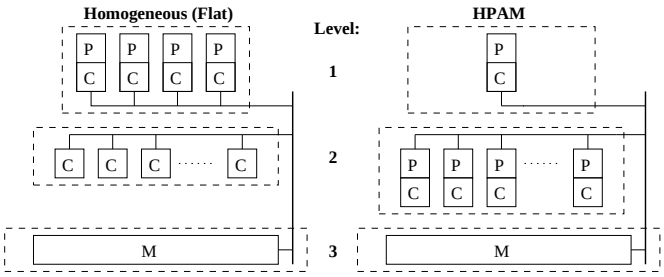

Figure 1: Homogeneous and HPAM architectures. P is a processor, **C** is cache memory, **M** is bulk memory

The following design parameters (summarized in Table 2 of Section 3), based on current technology values, have been assumed for both the flat and HPAM designs:

- **Processor performance:** A base processor with clock of 300MHz and CPI=1 is assumed for the first level of both the homogeneous and HPAM configurations. Following the cost-oriented approach on which the HPAM concept is based, the processors of the second level of the HPAM machine must be slower than the base processor. It is assumed that performance is proportional to the square root of cost [35]. Let FN_1 denote the number of processors with clock cycle $FCLK_1$ in the homogeneous machine, HN_2 the number of processors in the second level of the HPAM machine, and assume that the first level of HPAM has only one processor. The clock cycle of HPAM second-level processors is given by the following equation.

$$HCLK_2 = \sqrt{\frac{HN_2}{FN_1 - 1}} * FCLK_1 \quad (1)$$

- **Bus performance**: A bus clock of 60MHz and bus widths of both 16 and 32 Bytes are assumed. This choice is inspired by the use of a 66MHz, 16-Byte bus in Alpha 21164 [6] systems

- **Cache sizes:** The cache sizes assumed are based on miss rate measurements. The approach described in [25] is followed to obtain cache sizes that are scaled to the working set sizes of the benchmarks under study. Benchmarks from the CMU task-parallel [21], Perfect Club [15] and Spec95 [28] suites have been used in this study. The conclusion reached is that working sets for the CMU benchmarks are smaller than Perfect Club's, which are smaller than the Spec95 working sets. A cache that performs fairly well for a CMU benchmark is likely to perform very poorly on a Spec95 benchmark, while a cache large enough to perform well on a Spec95 benchmark may be larger than the entire data set of a CMU benchmark. Therefore, different cache sizes for these suites have been chosen. The configurations used are: direct-mapped L1 caches with 32Byte blocks, and 2-way set associative L2 caches with 64Byte blocks. The cache sizes shown in Table 1 correspond to the total cache available in the first and second levels of both HPAM and flat machines. The cache size of a processor in any given level equals the level cache size divided by the number of processors in that level.

- **Number of processors**: Since a parallel region of code is assumed to execute concurrently without overheads, small processor counts need to be used to minimize the error that this assumption introduces in the execution model. Homogeneous configurations with 4 and 16 processors, and HPAM configurations with one processor in the first level and 16 processors in the second level are used in this paper.

	Level 1 (KB)			Level 2 (MB)		
	CMU	Perf.	Spec	CMU	Perf.	Spec
HPAM	16	16	64	0.5	1	2
Flat	64	64	256	0.5	1	2

Table 1: Cache sizes assumed in the simulations

No assumption is made about the availability of components that rely on processor-memory integration and could improve HPAM performance (such as those recently proposed in [5, 24] and others likely to be available in the future). Instead a conservative form of computing-in-memory is assumed (perhaps better described as "computing-near-memory") that uses conventional processor and cache technologies. It is reasonable to expect that HPAM-intended processor-memory integrated components may lead to better performance than that observed in the simulations of the HPAM design used in this paper.

3 Performance Model

The performance model consists of expressions for execution times for both HPAM and homogeneous configurations based on the model parameters presented in Table 2 and Section 2. Figure 2 shows a graphical representation of the terminology used for access times across the hierarchy. The simulator is able to measure, for the instrumented benchmarks listed in Table 4, the parameters FIC_s, FIC_p, HIC_i, $FA_{1,j}$ and $HA_{i,j}$ of the execution time model.

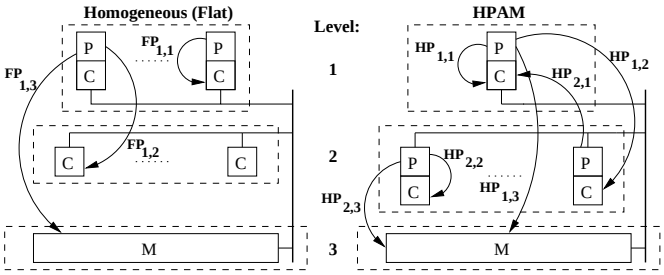

Figure 2: Parameters of the execution time model for 2-level HPAM and homogeneous configurations

The execution time expressions for the homogeneous and HPAM configurations are introduced in Equations 2 and 3, respectively. Equation 2 consists of three major terms. The first term is the time to execute the sequential fraction of the code, when only one processor is busy. It includes both instruction execution time and accesses to local memory. The second term is the time to execute the parallel fraction of the code, when all processors execute concurrently. The third term is the execution time contribution due to memory accesses to remote levels. It is assumed that the parallel fraction of the code can be executed without overheads due to synchronization and shared-memory traffic inside the level.

Equation 3 is similar to Equation 2, the main difference being that instructions and memory accesses can be issued in any level of the hierarchy that contains processors. Equation 3 assumes that the first HPAM level contains a single processor and handles the entire sequential fraction of the code (i.e., $HN_1 = 1$ and $HIC_1 = FIC_s$). It is also assumed that there is no overlap in execution between levels 1 and 2. For the 2-level configuration shown in Figure 2, $HIC_1 = FIC_s$, $HIC_2 = FIC_p$.

FN_1	number of processors in the flat machine
HN_i	number of processors in level i of HPAM
$FCLK_1$	clock cycle of the flat machine processors
$FCPI_1$	average clocks per instruction of processors of the flat machine
$HCLK_i$	clock cycle of level i HPAM processors
$HCPI_i$	average clocks per instruction of processors in level i of HPAM
$CSIZE_i$	total cache space available at level i
$BSIZE_i$	block size of the cache at level i.
DoP_t	minimum degree of parallelism for a DO loop to be considered parallel and be assigned to the second level of HPAM
FIC_s	number of instructions executed in the flat configuration in sequential mode
FIC_p	number of instructions executed in the flat configuration in parallel mode
HIC_i	number of instructions executed in level i of the HPAM configuration
$FAs_{1,j}$	number of memory references issued in the sequential code in the flat machine that are serviced by a cache in level j
$FAp_{1,j}$	number of memory references issued in the parallel code in the flat machine that are serviced by a cache in level j
$FA_{1,j}$	total number of memory references in the flat machine that are serviced by a cache in level j ($FA_{1,j} = FAs_{1,j} + FAp_{1,j}$)
$HA_{i,j}$	number of memory references issued in level i of HPAM that are serviced by a cache in level j
MAT_i	access time of the memory in level i
$FP_{1,j}$	when $j = 1$, $FP_{1,1}$ represents the hit time of the cache in the first level. For $j > 1$, $FP_{1,j}$ is the penalty of bringing a cache block from level j to 1 in the flat machine
$HP_{i,j}$	when $j = i$, $HP_{i,j}$ represents the hit time of the cache in the i^{th} level of HPAM. For $j \neq i$, $HP_{i,j}$ represents the penalty of bringing a cache block from level j to level i
CLK_{bus}	clock cycle of the inter-level bus
W_{bus}	width of the inter-level bus

Table 2: Notation used throughout the paper

$$FTime = \quad (2)$$
$$(FIC_s * FCPI_1 * FCLK_1 + FAs_{1,1} * FP_{1,1}) +$$
$$\left(\frac{FIC_p * FCPI_1 * FCLK_1 + FAp_{1,1} * FP_{1,1}}{FN_1}\right) +$$
$$\left(\sum_{j=2}^{levels} FA_{1,j} * FP_{1,j}\right)$$

$$HTime = \quad (3)$$
$$\left(\sum_{i=1}^{levels} \frac{HIC_i * HCPI_i * HCLK_i + HA_{i,i} * HP_{i,i}}{HN_i}\right) +$$
$$\left(\sum_{i=1}^{levels} \sum_{j=1, j\neq i}^{levels} HA_{i,j} * HP_{i,j}\right)$$

The determination of access times across the hierarchy ($FP_{1,j}$ and $HP_{i,j}$) is presented in the following discussion. Note that cache sizes affect indirectly the execution time equations, as different cache sizes generate different memory access distributions, captured by the parameters $FA_{1,j}$ and $HA_{i,j}$. For instance, a larger cache on level 1 is likely to increase both $FA_{1,1}$ and $HA_{1,1}$, i.e., the number of hits in the first level. The performance analysis concentrates on a 2-level HPAM configuration (Figure 2).

Memory access times: Memory access times need to be modeled for the caches on levels 1 and 2, and for main memory on level 3. The memory access times, together with the time to transfer a data block over the bus, are used to calculate the penalties $HP_{i,j}$ and $FP_{1,j}$. It is assumed that the on-chip L1 cache is accessed in two CPU clock cycles, as in the Alpha 21164 [6]. For the second-level cache, it is assumed that the same SRAM technology of the L1 cache is used. The memory access time has to be scaled according to the size difference between the two caches. The assumption used is that the second-level cache is divided into HN_2 banks, one bank per processor on the second level of the HPAM machine. Hence, each processor on the second level of HPAM has an on-chip cache of size $\frac{CSIZE_2}{HN_2}$. According to [27], the access time increase due to a larger on-chip cache size can be approximated by a constant factor when the cache size is doubled. This constant factor is of about 14% for a 0.8μm technology [27]; this value is assumed in the performance analysis to obtain an expression for MAT_2 in terms of MAT_1 and the cache sizes. Equation 4 models a compound 14% increase in access time for each doubling of the on-chip cache size.

$$MAT_2 = MAT_1 * (1.14)^{\log_2 \frac{(CSIZE_2/HN_2)}{CSIZE_1}} \quad (4)$$

Penalties $FP_{1,j}$, $HP_{i,j}$: There are two cases to be considered:

1. *Hit times ($i = j$):* As Figure 2 depicts, it is assumed that processors and memory (cache) are implemented in the same die. There is no need to access off-chip data on a hit, and the assumption made is that the width of the on-chip memory data-path is as large as a cache line. Given these assumptions, the hit time on level i is set to the access time of the memory on that level. For the homogeneous configuration, it follows that $FP_{1,1} = MAT_1$, and for the HPAM configuration it follows that $HP_{1,1} = MAT_1$ and $HP_{2,2} = MAT_2$

2. *Miss penalties ($i \neq j$):* For the homogeneous configuration, it is assumed that the miss penalty is given by the access time of the memory providing the data to serve the miss, plus transfer time on the bus.

The argument for computing-in-memory is built upon the fact that off-chip accesses are expensive: the off-chip available bandwidth is smaller than the on-chip bandwidth due to packaging constraints (limited number of pins), and the latency is larger due to the necessity of driving off-chip lines with larger capacitance [5, 24]. For the configurations shown in Figure 2, the possible performance gains of an HPAM over a flat machine due to computing-in-memory are based on the fact that an access generated by an HPAM level-2 processor that hits in its local cache is faster than an access generated by a level-1 processor on the homogeneous configuration that misses on its local cache and requires the crossing of the chip boundaries to fetch data from the lower levels.
The off-chip overhead is modeled by assuming a bus that is narrower than a block size and whose clock is slower than the on-chip clock. An off-chip memory access thus requires one bus transaction to request data and one or more bus cycles to transfer a cache line, in addition to the memory access time. Equations 5 and 6 show the resulting model for the penalties for both homogeneous and HPAM configurations. It is assumed that the miss penalty $HP_{1,2}$ of the HPAM machine is greater than or equal to the corresponding penalty $FP_{1,2}$ of the homogeneous machine. The non-negative parameter OH_{PAM} represents potential overheads of accessing memory on a processor-and-memory chip compared to accessing a memory-only chip. Contention on the PAM memory ports when both the on-chip processor and an external processor request a memory access is thus captured by OH_{PAM}. This parameter is varied in the simulations in order to observe its impact on performance.

$$FP_{1,2} = MAT_2 + (1 + \frac{BSIZE_1}{W_{bus}}) * CLK_{bus} \tag{5}$$

$$HP_{1,2} = FP_{1,2} + OH_{PAM} \tag{6}$$

Table 3 shows the actual timings used in the simulations of CMU and Spec95 benchmarks for the parameters modeled in Equations 4, 5 and 6 (timings for the Perfect Club suite are slightly different due to a different ratio of level 2 and level 1 cache sizes). The range shown for $HP_{1,2}$ corresponds to OH_{pam} ranging from 0 to 50 base processor clock cycles.

W_{bus}	MAT_1	MAT_2	$FT_{1,2}$	$FP_{1,2}$
16B	6.7ns	12.8ns	63ns	63ns - 230ns
32B	6.7ns	12.8ns	46ns	46ns - 213ns

Table 3: Access times used in simulations of CMU and Spec95 benchmarks

As for miss penalties to the main memory (level 3), a fixed value of 300ns is assumed for both configurations based on the main memory latency of an Alpha server [5, 36]: $FP_{1,3} = HP_{1,3} = 300ns$. It remains to model $HP_{2,1}$ and $HP_{2,3}$ for the HPAM architecture. It is assumed that the penalty to bring data from the first level to the second level is as large as the level-2 penalty seen by the first level, i.e, $HP_{2,1} = HP_{1,2}$, and that the memory penalty is the same regardless of the access being generated on the first or second level of HPAM, i.e., $HP_{2,3} = HP_{1,3}$.

4 Evaluation Methodology

The benchmarks studied in this paper are listed in Table 4, together with the total number of memory references generated during their simulation.

All the benchmarks studied are originally sequential and written in Fortran. Each benchmark is instrumented with Polaris [29], a source-to-source parallel compiler, to identify parallel DO-ENDDO loops. Each identified parallel loop is "wrapped" with traps to the simulator engine at the beginning and end of

Suite	Benchmark	# Mem. Refs.
CMU	Airshed	4.47E9
CMU	Stereo	7.39E8
CMU	Radar	3.14E8
CMU	FFT2	6.17E8
Perfect Club	Ocean	2.15E10
Perfect Club	Arc2D	2.03E10
Perfect Club	FLO52	5.30E9
Spec95	Hydro2d	1.05E11
Spec95	Tomcatv	7.90E10

Table 4: Benchmarks used in the simulation analysis

the loop. The trap signals the engine that a parallel region of the code with a specific degree of parallelism has been reached.

The simulator is composed of an execution-driven engine and a multi-level cache simulator built on top of Shade [1]. The simulator engine is invoked on each trap, instruction and data reference issued by the benchmark under study.

The memory hierarchy simulator implements a data cache for each level of an HPAM configuration. Each cache is defined by its size, block size and associativity. Block sizes in a given level of the hierarchy must be either of the same size as a block of the upper level cache, or larger by a power-of-two factor. The cache coherence protocol used is based on the MESI [16] protocol with extensions to support memory accesses being potentially generated on every level of the hierarchy [31]. Instruction caches are assumed to be perfect (100% hit rate) by the simulator.

The memory hierarchy simulator has been validated by comparing its output with results from Cachesim5 (a cache analyzer distributed with Shade) for the benchmarks under study, with identical data cache configurations (number of levels, cache and block sizes and associativity of each level) and with all references being generated in the first level of the HPAM hierarchy.

5 Results and Analysis

In this section, the data obtained from simulations of the benchmarks listed on Table 4 is presented and analyzed, assuming a 2-level HPAM configuration. Initially, the existence of level locality of references on an HPAM design is exposed. Two design scenarios are then introduced to serve as a basis to the performance comparison between HPAM and flat machines, and the impacts of computing-in-memory and heterogeneity on HPAM's performance are analyzed.

5.1 Benchmark locality and miss rates

Table 5 presents the basic characteristics of the benchmarks studied: total instructions count (IC), degree of parallelism threshold used to determine parallel loops (DoP_t), fraction of instructions that are executed in sequential mode (IC_s), and local hit rates for levels 1 and 2 of the HPAM configuration. The hit rate of HPAM level L_i shown in Table 5 is defined as the number of data references issued by processors in level i that hit in level i divided by the total number of references issued in level i, as shown in Equation 7.

$$Hit, L_i = \frac{HA_{i,i}}{\sum_{j=1}^{levels} HA_{i,j}} \tag{7}$$

Bench.	IC	DoP_t	IC_s %	L1 hit %	L2 hit %
Airshed	6.49E10	25	1.7%	93.3%	99.8%
Stereo	3.64E9	256	23.0%	97.4%	98.4%
Radar	2.63E9	400	97.4%	96.6%	99.7%
FFT2	3.59E9	64k	86.5%	99.4%	99.8%
Ocean	1.88E11	64	89.9%	98.9%	99.9%
Arc2D	1.49E11	16	0.04%	97.1%	99.5%
FLO52	3.10E10	16	1.2%	96.5%	99.9%
Hydro2d	4.13E11	64k	1.2%	97.0%	99.4%
Tomcatv	2.65E11	511	7.0%	98.4%	99.4%

Table 5: Benchmark summary: number of instructions, minimum degree of parallelism executed in the second level of HPAM (DoP_t), % sequential execution and L1 and L2 hit rates for the HPAM configuration

The values of DoP_t shown in Table 5 correspond to the degree of parallelism threshold used to decide whether a DO loop is executed in parallel, in the second level of HPAM, or sequentially, in the first level. Except for *Flo52*, DoP_t is also used to decide whether a loop is executed in parallel or sequentially in the flat machine. The value of DoP_t is chosen so that the percentage of code with parallelism between 1 and DoP_t is either zero or negligible. As an example, there is no loop in *Stereo* satisfying $1 < DoP < 256$. In *Hydro2d* only 0.19% of the instructions are executed in loops satisfying $1 < DoP < 64k$. *Flo52* is the only exception from this set of applications where there is a significant fraction of the code with degree of parallelism between 1 and 16 ($DoP = 8$ in *Flo52*). For Scenario 1, the results account for the fact that such fraction of code executed in parallel by four processors in the flat machine and serially by the HPAM machine.

The level hit rates shown in Table 5 show that the applications under study exhibit good locality with respect to degree of parallelism [35]. A more detailed look at the distribution of accesses across the hierarchy is presented in Table 6 for the benchmark Airshed. The total number of accesses is the same for both HPAM and flat machines; the way accesses are divided across the levels, however, is different.

The access distribution shows how accesses generated by the i^{th} HPAM level tend to be locally served (values shown in bold face). In this example, the sum of HPAM level-1 and level-2 hits ($HA_{1,1} + HA_{2,2}$) is larger than the number of flat level-1 hits ($FA_{1,1}$). In addition, the number of HPAM level-1 misses serviced by level 2 ($HA_{1,2}$) is about 3.3 times smaller than the corresponding number of misses on the flat machine ($FA_{1,2}$). This means that the HPAM processors hit their level caches more often than the flat design.

The level locality behavior observed on Tables 5 and 6 confirms and extends results obtained on early studies on the HPAM architecture for realistic cache configurations. In this paper, the performance impact of the combination of computing-in-memory and level locality is studied.

Consider the number of accesses on level 1 of the flat configuration that miss the level 1 cache and need to be serviced by the second level cache. In an HPAM configuration, many of these accesses will be issued by processors in the second level, and consequently are likely to be satisfied by an on-chip cache. This corresponds to a second level hit, which is faster than a miss to the second level on the homogeneous architecture, since there is no off-chip overhead involved.

5.2 Speedup analysis

The approach taken to study the performance benefits of computing-in-memory consists of comparing a 2-level HPAM machine to a homogeneous machine. The basis of comparison is execution time, as modeled in

Flat		
$FA_{1,1}$	$FA_{1,2}$	$FA_{1,3}$
4,421,280,188	43,643,411	3,604,499
HPAM		
$HA_{1,1}$	$HA_{1,2}$	$HA_{1,3}$
209,486,013	13,417,183	1,635,899
$HA_{2,1}$	$HA_{2,2}$	$HA_{2,3}$
6,561,554	**4,235,536,996**	1,890,453

Table 6: Distribution of accesses across levels for the benchmark Airshed on both 2-level HPAM ($HA_{i,j}$) and flat ($FA_{1,j}$) configurations.

Section 3. The two architectures are compared under two different scenarios. The choice of the scenarios is based on a *constant-cost* approach: a base homogeneous machine with four identical processors is assumed, and two different designs with equivalent cost are generated by following the cost/performance model of Equation 1:

- **Scenario 1:** The homogeneous machine has 4 identical processors, each clocked at 300MHz and with a CPI of one. The HPAM machine has 1 processor on the first level identical to the processor used on the homogeneous machine; this processor is responsible for executing the sequential fractions of the application. The second level contains 16 processors, each of them clocked at 129MHz (value obtained from Equation 1). This scenario is based on the HPAM concept that fast processors are expensive and should be used in sequential computation while parallel computation should be handled by a larger number of slower processors.

- **Scenario 2:** The HPAM machine is identical to the one described for Scenario 1. However, the homogeneous machine has sixteen processors instead of four. Hence both HPAM and flat machines execute parallel code with the same number of processors. The same principle used in the square-root cost model of Equation 1 is used to determine the clock speed of each processor based on the base clock speed of 300MHz: with the same cost of four 300MHz processors, it is possible to implement sixteen processors clocked at $\sqrt{4/16} * 300 = 150$MHz. Note that if a CPI of one is assumed for all processors and the performance of the processors in terms of MIPS is combined, on the flat machine it follows that the peak performance is 16*150=2400 MIPS, while on the HPAM machine the peak performance is 300+16*129=2364 MIPS. So, the peak performances of both machines are equivalent, and performance differences between the two architectures are due to the heterogeneity and computing-in-memory capabilities of the HPAM configuration.

The simulation results presented in the rest of this section refer to these two scenarios. For a given scenario, the simulation plots show speedup, defined as execution time on the homogeneous machine divided by execution time on the HPAM machine, for varying PAM overhead OH_{pam}, with the bus width assuming values of 16 and 32 Bytes. The benchmark *Tomcatv* is first examined. Figure 3 shows the speedup of HPAM over a homogeneous configuration assuming Scenario 1. In this case, HPAM performs 34% to 42% better than the homogeneous case for the two bus widths considered and for OH_{pam} up to 50 clock cycles of the base CPU shown in the graph. Notice that for a bus width of 32 Bytes (dotted line of Figure 3), a level-1 cache block can be fetched to the cache in one bus cycle. Even with a wide bus, the time to access the level-2 cache from level 1 in the homogeneous configuration is larger than the time to access the level-2 cache from level 2 in the HPAM case, since the off-chip datapath is clocked at a slower rate than the on-chip datapath.

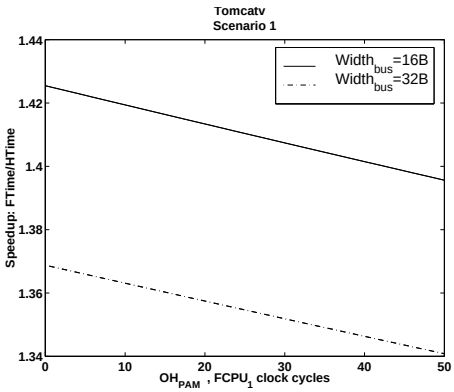

Figure 3: Speedup curve for Tomcatv, Scenario 1. Speedup is the ratio of the execution times, and the X axis is the PAM overhead modeled on Equation 6. Solid and dashed lines correspond to bus widths of 16 and 32 Bytes, respectively.

Figure 4 shows the speedup curves for the benchmark *Tomcatv*, now assuming Scenario 2. In this scenario, the homogeneous configuration is capable of executing the parallel fraction of the code faster than the HPAM machine can, because each homogeneous processor is 16% faster than the HPAM processors on the second level. However, the serial fraction of the code is executed in the homogeneous machine at 50% of the speed of the HPAM level-1 processor. As *Tomcatv* has about 7% of its execution in sequential mode (see Table 5), the time spent in the sequential fraction of the code is significant in the total execution time. As HPAM executes this portion of the code twice as fast as the homogeneous machine, the total execution time is 4% to 11% smaller for HPAM.

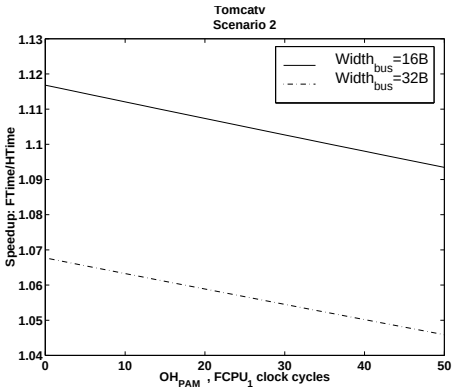

Figure 4: Speedup curve for Tomcatv, Scenario 2.

For an application with more available parallelism, *Airshed* (Figure 5), HPAM performs worse than the homogeneous configuration in Scenario 2. In this scenario, the homogeneous configuration executes the parallel fraction of the code faster than HPAM, and as the execution of the sequential fraction (about 1.7% of the code) is no longer the dominant factor in the performance, the overall execution time is smaller in the homogeneous case. However, HPAM performs significantly better for the same application if Scenario 1 is assumed, as shown in Table 7.

For an application with most of its execution in sequential mode, *Radar*, HPAM performs worse than the homogeneous machine for scenario 1 if the overhead OH_{pam} increases, as shown in Figure 6. In this case, the second level of the HPAM machine is used rarely and the possible performance improvement

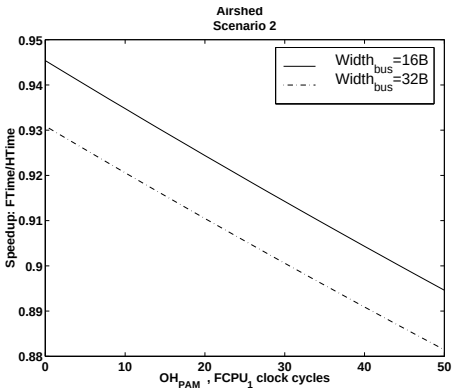

Figure 5: Speedup curve for Airshed, Scenario 2.

from computing-in-memory is not sufficient to offset the increase in the miss penalty $HT_{1,2}$ due to OH_{pam}. However, HPAM performs better than the homogeneous machine for scenario 2 (see Table 7) since the sequential code is executed faster.

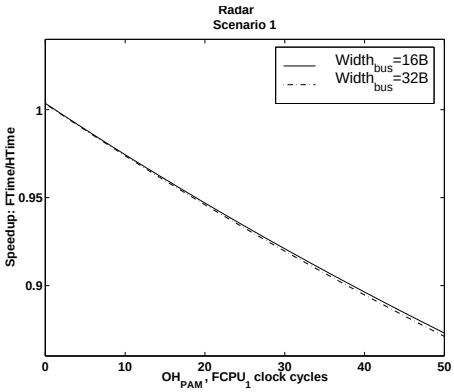

Figure 6: Speedup curve for Radar, Scenario 1.

Table 7 summarizes the results for the benchmarks that have been studied for both scenarios, and assuming a block size of 32 Bytes, OH_{pam} =0 and 50. The HPAM configuration outperforms the homogeneous configuration in Scenario 1 except for the cases where the application is highly sequential. Two reasons contribute to this performance advantage:

1. The larger number of processors in the HPAM configuration. By following a constant-cost approach, assuming the quadratic cost/performance rule shown in Equation 1, the second level of the HPAM system has a larger aggregate performance than the homogeneous machine.

2. Good locality of references with respect to degree of parallelism, combined with computing-in-memory in the second level. Good locality is responsible for keeping the inter-level communication low, while computing-in-memory provides fast access to on-chip memory for processors in the second level.

As for Scenario 2, HPAM outperforms the homogeneous configuration when the fraction of the code that is sequential is significant. This conclusion had been reached previously when the benchmark *Tomcatv* was analyzed (Figure 4). When the application is highly parallel, the homogeneous machine outperforms HPAM because each processor in the homogeneous machine is faster than an HPAM second-level processor,

Bench.	Scenario 1, OH_{pam} =		Scenario 2, OH_{pam} =	
	0	50	0	50
Airshed	1.63	1.51	0.96	0.89
Stereo	1.08	1.01	1.19	1.11
Radar	1.00	0.87	1.78	1.54
FFT2	1.01	0.98	1.70	1.64
Ocean	1.01	0.97	1.81	1.73
Arc2d	1.64	1.63	0.92	0.92
FLO52	1.70	1.66	0.89	0.88
Hydro2d	1.55	1.53	1.01	1.00
Tomcatv	1.37	1.34	1.07	1.05

Table 7: Summary of results: Speedups for scenarios 1 and 2, and PAM overheads of 0 and 50 base CPU clock cycles

and because the fast processor on the first level of HPAM is idle during the parallel execution in the second level. It is conceivable that more sophisticated versions of HPAM might be able to also use the fast processor for parallel computation and eliminate this disadvantage.

In summary, for applications with significant available parallelism, the use of a larger number of slower processors for parallel execution and of a single fast processor for sequential execution on HPAM yields performance improvements over a flat machine with fast, but fewer, processors. For significantly sequential applications, the use of a single fast processor and several slower processors on HPAM yields performance improvements over a flat configuration with good parallel performance but poor sequential performance.

5.3 Small cache analysis

The results shown in Tables 5 and 7 are relative to the assumed cache configurations for the three different benchmark suites, presented in the previous section. Simulations in some of the benchmarks with cache sizes (levels 1 and 2) four times smaller than the assumed values have also been performed. When smaller L1 caches are used, the HPAM execution time increases at a higher rate as OH_{pam} increases, since more level-1 references are likely to miss on the first HPAM level and be serviced by the second level. For very small L1 caches the level-1 miss rate can be high for HPAM. Because HPAM has fewer level-1 processors, it has effectively a smaller amount of cache available in the first level than the flat configuration, since it is assumed that each level-1 processor is identical and has the same amount of on-chip cache. Taking the benchmark *Airshed* as an example, using 2KByte as the on-chip L1 cache size and 64KByte as the L2 cache size, HPAM shows a very high level-1 miss rate of about 39%; even with such high miss rate, HPAM performs better than the flat machine for the first scenario, since the application is very parallel and the level-2 HPAM miss rate is relatively low (0.6%). For such cache configuration the speedup entries for *Airshed* on table 7 would change to 1.57 and 1.21 for Scenario 1, OH_{pam} =0 and 50, respectively.

5.4 Impact of the memory wall

Scenarios 1 and 2 have been so far analyzed with the execution time model parameter values based on current technology. A motivation for the design of HPAM machines is the observation that the performance gap between processors and memory has been widening as the rate at which processor speed improves is greater than the rate at which memory speed increases [30]. Consider a scenario where the homogeneous configuration shown in Figure 1 has 2 processors on the first-level and the HPAM has one processor on

the first level and two processors on the second level (i.e., $HN_1 = 1$ and $FN_1 = HN_2 = 2$). Suppose that the HPAM first-level processor is identical to the processor used in the homogeneous machine, and that the clock cycle of the second-level HPAM processors is twice the cycle of the first-level processor (i.e., $FCLK_1 = HCLK_1$ and $HCLK_2 = 2 * HCLK_1$). It is again assumed that there is no overlap in execution between the HPAM levels. The parallel performance of the flat machine is thus twice the parallel performance of the HPAM machine, while their sequential performances are equivalent.

Assuming the same initial values used in this paper for processor and memory speeds, Figure 7 shows how the execution times for both configurations vary for the benchmark *Hydro2d* when: clock rate and on-chip memory speed increase at a rate of 60%; main memory speed increases at a rate of 10% and bus speed remains constant.

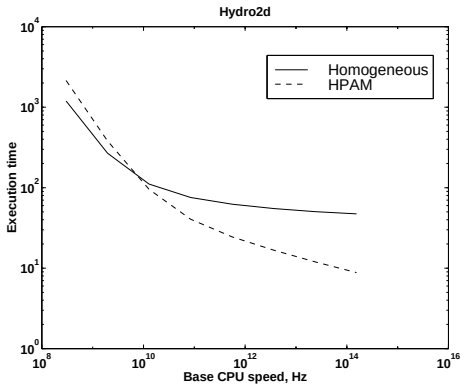

Figure 7: Execution times for homogeneous and HPAM configurations when clock speed is increased and bus speed is fixed.

The application *Hydro2d* spends only about 1.2% of its execution time in sequential mode. The remaining fraction of the code is executed by the flat machine twice as fast as HPAM when the off-chip overhead is comparable to processor speed. However, when the time to service off-chip accesses between levels 1 and 2 becomes very large compared to the processor speed, the parallel execution time of the flat machine is dominated by the off-chip overhead. If clock speed keeps increasing, the homogeneous machine is not able to reduce the execution time as effectively as HPAM does, and the latter begins to outperform the homogeneous machine.

6 Intra-level Data Movement

The execution time model used throughout this paper has been designed to allow simulation of the applications under study within acceptable time. It is intended to provide estimates of execution times that allow a performance comparison between HPAM and homogeneous machines rather than give highly accurate results.

The execution model does not include the effects of intra-level data movement. Several factors suggest that the relative impact of this simplification is small:

- The parallel sections of the benchmarks consist of either independent loops (i.e. without data dependencies) or reductions; hence communication (other than that due to poor data distribution) is either minimal or can be optimized for efficiency; syncronization traffic is restricted to end-of-loop barriers.
- The multiprocessors under study have relatively few processors, thus keeping the probability of poor data distribution small and allowing the assumption of fast inter-processor communication.

- When contrasting the behavior of multiprocessors of similar sizes the effects of intra-level communication on each machine are likely to cancel each other when computing relative performance.

In spite of the above comments, a simplified intra-level communication model has been added to the execution time model. The assumptions used in the intra-level communication model are the following:

- Intra-level miss rates are introduced in the model for both configurations. It is assumed that a fraction of *(1-miss_rate)* of the intra-level hits are to local, on-chip cache, while the remaining accesses are serviced by another cache in the same level.

- Intra-level miss penalties are also introduced. The product of intra-level memory references, miss rate and miss penalty adds to the average intra-level memory access time.

The intra-level miss penalties have been set to the inter-level penalty between levels one and two (i.e., $HP_{1,2}$ for HPAM and $FP_{1,2}$ for the flat machine) in the following analysis. Intra-level miss rates are varied from 0% to 50% for the second level of HPAM. The relationship between level-2 HPAM intra-level miss rates and level-1 flat miss rates is complex, since the on-chip cache sizes and number of processors are different. It is conservatively assumed in the following analysis that the intra-level miss rate of the first level of the flat machine is RP times smaller than HPAM's second level miss rate, where RP is the ratio between number of processors in the two levels (i.e., $RP = HN_2/FN_1$). For Scenario 1, we have that $RP = 16/4 = 4$, i.e., the flat intra-level miss rate is one quarter of the HPAM intra-level miss rate.

Table 8 shows the speedup of HPAM versus a flat machine when this intra-level communication model is applied to the results previously obtained for Scenario 1. For each intra-level miss rate, two values of speedup are shown, corresponding to ratios of HPAM and flat intra-level miss penalties of $HP_{1,2}/FP_{1,2} = 2.7$ (when $OH_{pam} = 25$) and $HP_{1,2}/FP_{1,2} = 4.6$ (when $OH_{pam} = 50$).

Bench.	OH_{pam}	HPAM intra-level miss rate			
		0%	10%	25%	50%
Airshed	25	1.57	1.49	1.38	1.24
	50	1.51	1.37	1.21	1.02
Stereo	25	1.05	1.02	0.98	0.92
	50	1.01	0.95	0.88	0.79
Arc2d	25	1.64	1.49	1.32	1.13
	50	1.64	1.38	1.13	0.89
Flo52	25	1.68	1.47	1.26	1.04
	50	1.66	1.31	1.02	0.78
Hydro2d	25	1.54	1.38	1.21	1.02
	50	1.53	1.26	1.01	0.79
Tomcatv	25	1.36	1.24	1.11	0.96
	50	1.34	1.14	0.95	0.76

Table 8: HPAM speedup assuming Scenario 1, intra-level miss rates from 0% to 50% and PAM overheads of 25 and 50 base CPU clock cycles.

Both machines slow down when the intra-level miss rate is modeled. However, Table 8 shows that HPAM continues to perform better than the flat machine even with larger intra-level miss rates and penalties. The assumptions of intra-level penalties and miss rates made in this analysis are very conservative, since intra-level communication is assumed to be as expensive as inter-level communication, and HPAM miss rates are assumed to be very large.

7 Conclusions

The collected experimental data shows that applications exhibit good locality of reference with respect to degree of parallelism. An HPAM machine benefits from such locality property in several ways. First, communication between HPAM levels is reduced, since most of the references issued from a processor tend to be serviced by a processor in the same level. In addition, computing-in-memory also benefits from high level locality, since processors in the lower hierarchy levels have fast access to local, on-chip data.

The performance analysis presented in this paper has shown that the combination of heterogeneity, locality with respect to degree of parallelism and computing-in-memory results in better execution times for a 2-level HPAM system when compared to a homogeneous design for several applications with different parallelism profiles, under a constant cost model.

Design parameters based on current technology have been assumed throughout this paper along with rather conservative assumptions. These assumptions might be justifiable in light of engineering proposals to reduce the processor/memory speed gap such as address pipelining/interleaving techniques and new RAMBUS-like technologies. This paper shows that the performance advantages of an HPAM architecture are already evident with current technology. It is believed that they can only become more significant with increasing processor/memory speed gap and deeper memory hierarchies.

References

[1] B. Cmelik and D. Keppel. Shade: A fast instruction-set simulator for execution profiling. In *Proceedings of the 1994 SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, 1994.

[2] Cray Research. *Cray T3D System Architecture Overview.* Cray Research Inc., Tech. Rep. HR-04033, September 1993.

[3] D. Burger. System-Level Implications of Processor-Memory Integration. In *Workshop on Mixing Logic and DRAM: Chips that Compute and Remember, ISCA '97*, June 1997.

[4] D. Menasce and V. Almeida. Cost-performance Analysis of Heterogeneity in Supercomputer Architectures. *Proc. Supercomputing Conf.*, November 1990.

[5] D. Patterson et al. A Case for Intelligent RAM: IRAM. *IEEE Micro*, Apr 1997.

[6] Digital Equipment Corporation. *Alpha 21164 Microprocessor Hardware Reference Manual*, 1994.

[7] E.E. Johnson. Graffiti in the "Memory Wall". *ACM Computer Architecture News*, 1995.

[8] C.B. Stunkel et al. The SP2 High-Performance Switch. *IBM Systems Journal*, 34(2), 1995.

[9] D. Moncrieff et al. Heterogeneous Computing Machines and Amdahl's Law. *Parallel Computing*, 22(3), 1996.

[10] H.A. Grosch. High Speed Arithmetic: The Digital Computer as a Research Tool. *J. Opt. Soc. Am.*, 43, Apr 1953.

[11] H.A Grosch. Grosch's Law Revisited. *Computerworld*, Apr 1975.

[12] J.B. Andrews and C.D. Polychronopoulos. An analytical approach to performance/cost modeling of parallel computers. *Par. and Dist. Computing*, 12, 1991.

[13] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 1996.

[14] L. Gwennap. Estimating IC Manufacturing Costs. *Micropocessor Report*, August 1993.

[15] M. Berry et al. The Perfect Club Benchmarks: Effective Performance Evaluation on Supercomputers. Technical Report UIUC-CSRD-827, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, July 1994.

[16] M. Papamarcos and J. Patel. A Low Overhead Coherence Solution for Multiprocessors with Private Cache Memories. In *Proc. of 11th Annual Int. Symp. on Computer Architecture*, 1984.

[17] M.L. Barton and G.R. Whiters. Computing Performance as a Function of the Speed, Quantity, and Cost of the Processors. *Supercomputing*, 1989.

[18] M.V. Wilkes. The Memory Wall and the CMOS End-Point. *ACM Computer Architecture News*, 1995.

[19] N. Yamashita et al. A 3.84GIPS Integrated Memory Array Processor LSI with 64 Processing Elements and 2Mb SRAM. In *Proceedings, ISSCC*, 1994.

[20] N.P. Jouppi and P. Ranganathan. The Relative Importance of Memory Latency, Bandwidth, and Branch Limits to Performance. In *Workshop on Mixing Logic and DRAM: Chips that Compute and Remember, ISCA '97*, June 1997.

[21] P. Dinda et al. The CMU Task parallel Program Suite. Technical Report CMU-CS-94-131, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, Jan 1994.

[22] P. Ein-Dor. Grosch's Law Re-Revisited: CPU Power and the Cost of Computation. *Communications of the ACM*, 28(2), Feb 1985.

[23] P.M. Kogge and T. Sterling and G. Gao. Processing In Memory: Chips to Petaflops. In *Workshop on Mixing Logic and DRAM: Chips that Compute and Remember, ISCA '97*, June 1997.

[24] P.M. Kogge and T. Sunaga and H. Miyataka and K. Kitamura and E. Retter. Combined DRAM and Logic Chip for Massively Parallel Systems. *16th Conference on Advanced Research in VLSI*, 1995.

[25] S.C. Woo et al. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd Annual Int. Symp. on Computer Architecture*, July 1995.

[26] Semiconductor Industry Association. The National Technology Roadmap for Semiconductors. San Jose, CA, 1994.

[27] S.J.E. Wilton and N.P. Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. Technical Report WRL Research Report 93/5, Western Research Laboratory, Digital Equipment Corporation, 1993.

[28] Standard Performance Evaluation Corporation. Spec newsletter, Sep 1995.

[29] W. Blume, R. Doallo, R. Eigenmann, J. Grout, J. Hoeflinger, T. Lawrence, J. Lee, D. Padua, Y. Paek, B. Pottenger, L. Rauchwerger and P. Tu. Parallel Programming with Polaris. *IEEE Computer*, Dec 1996.

[30] W.A. Wulf and A.A. McKee. Hitting the Memory Wall: Implications of the Obvious. *ACM Computer Architecture News*, 1995.

[31] Z. Ben-Miled. A Hierarchical Solution to High Performance General Purpose Computing. Preliminary thesis report, Purdue University, 1995.

[32] Z. Ben-Miled and J.A.B. Fortes. A Heterogeneous Hierarchical Solution to Cost-efficient High Performance Computing. *Par. and Dist. Processing Symp.*, Oct 1996.

[33] Z. Ben-Miled, J.A.B. Fortes, R. Eigenmann, and V. Taylor. A Simulation-based Cost-efficiency Study of Hierarchical Heterogeneous Machines for Compiler and Hand Parallelized Applications. *9th Int. Conf. on Par. and Dist. Computing and Systems*, Oct 1997.

[34] Z. Ben-Miled, J.A.B. Fortes, R. Eigenmann, and V. Taylor. Towards the Design of a Heterogeneous Hierarchical Machine: A Simulation Approach. *30th Simulation Symp.*, April 1997.

[35] Z. Ben-Miled, R. Eigenmann, J.A.B. Fortes, and V. Taylor. Hierarchical Processors-and-Memory Architecture for High Performance Computing. *Frontiers of Massively Parallel Computation Symp.*, Oct 1996.

[36] Z. Cvetanovic and D.D. Donaldson. AlphaServer 4100 Performance Characterization. Technical report, Digital Equipment Corporation, 1997.