**Zina Ben Miled** received the B.S degree in Electrical Engineering from Oregon State University in 1988, the M.S. and Ph.D. degree from Purdue University in 1990 and 1997, respectively. She is currently a visiting assistant professor in the Electrical Engineering Department of the Purdue School of Engineering at Indianapolis. She is a member of the Institute of Electrical and Electronics Engineers (IEEE). Her research interests include parallel processing and computer architecture.

**Jose A.B. Fortes** received the B.S. degree in Electrical Engineering (Licenciatura em Engenharia Electrotecnica) from the Universidade de Angola in 1978, the M.S. degree in Electrical Engineering from the Colorado State University, Fort Collins in 1981 and the Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles in 1984. In 1984, he joined the faculty of the School of Electrical Engineering at Purdue University, West Lafayette, Indiana, where he currently is a Professor. From July 1989 through July 1990 he served at the National Science Foundation as director of the Microelectronics Systems Architecture program. From June 1993 till January 1994 he was a Visiting Professor at the Computer Architecture Department of the Universitat Politecnica de Catalunya in Barcelona, Spain. His research interests are in the areas of parallel processing, computer architecture, network computing and fault-tolerant computing. He has authored or coauthored over 90 technical papers. His research has been funded by the Office of Naval Research, AT&T Foundation, IBM, General Electric and the National Science Foundation.

Fortes is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE) professional society. He was a Distinguished Visitor of the IEEE Computer Society from 1991 till 1995. Fortes is/was on the Editorial Boards of the *Cluster Computing: The Journal of Networks, Software Tools and Applications*, the *IEEE Transactions on Parallel and Distributed Systems*, the *International Journal on Parallel Programming*, the *Cluster Computing: The Journal of Networks, Software Tools and Applications*, the *Journal of VLSI Signal Processing* and the *Journal of Parallel and Distributed Computing*.

**Rudolf Eigenmann** received his Ph.D. in Electrical Engineering / Computer Science 1988 from ETH Zurich, Switzerland. From 1988-1995 he was a member of the research staff at the Center for Supercomputing Research and Development (CSRD), University of Illinois at Urbana-Champaign. In 1995 he joined the faculty at the School of Electrical and Computer Engineering at Purdue University. He has worked in the areas of compilers, languages and tools for parallel machines; in performance evaluation; and in the design of high performance computer architectures. He is a member of the editorial board of the International Journal of Parallel Programming and IEEE Computational Science and Engineering. He also currently serves as the chairman of the High-Performance Group of the Standard Performance Evaluation Corporation (SPEC).

[10] J.B. Andrews and C.D. Polychronopoulos, An analytical approach to performance/cost modeling of parallel computers. *Par. and Dist. Computing*, 12, 1991, 343–356.

[11] S.Q. Moore and L.M. Ni, The effects of Network Contention on Processor Allocation Strategies. *IPPS*, 1996, 268-273.

[12] T.H. Dunigan, *Early Experiences and Performance of the Intel Paragon*. Oak Ridge National Laboratory, Tech. Rep. TM-12194, 1994.

[13] D. Basak and D.K. Panda, Designing Large Hierarchical Multiprocessor Systems under Processor, Interconnection, and Packaging Constraints. *ICPP*, I, 1994, 63–66

[14] J.J. Dongarra and T.H. Dunigan *Message-Passing Performance of Various Computers*. Oak Ridge National Laboratory, Tech. Rep. TM-13006, 1996.

[15] T. von Eicken, D.E. Culler et. al., Active Messages: a Mechanism for Integrated Communication and Computation. *ISCA*, 1992.

[16] C. Dubnicki, L. Iftode et. al., Software Support for Virtual Memory-Mapped Communication. *IPPS*, 1996, 372–381.

[17] Intel Supercomputer Systems Division. *Paragon System Overview*. Intel Corp., Santa Clara, CA, 1994.

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

[19] C.B. Stunkel, D.G. Shea et. al., The SP2 High-Performance Switch. *IBM Systems Journal*, 34(2), 1995, 185–203.

[20] K.P. Belkhale and P. Banerjee, A Scheduling Algorithm for Parallelizable Dependent Tasks. *IPPS*, 1991.

[21] Z. Ben-Miled, J.A.B. Fortes, R. Eigenmann and V. Taylor, Efficient Implementation of Unicast Based Broadcast, Gather and Scatter in a Two-Level Heterogeneous Architecture. *Hawaii Int. Conf. on System Sciences*, 1998, 216–225.

[22] R.J.O. Figueiredo, J.A.B. Fortes, Z. Ben Miled, V. Taylor and R. Eigenmann, *Impact of Computing-in-Memory on the Performance of Processor-and-Memory Hierarchies*, Tech. Rep. TR-ECE-98-1, Purdue Univ., 1998.

(this machine organization is indicated by "L" in Table 3). These two benchmarks have a low CCratio. Furthermore, the two benchmarks that can benefit from an HPAM machine with a high ratio of processor speed between levels are Cfft2 and Hairshed. These two benchmarks have a high CCratio. In general, the performance of a given benchmark when executed on a multilevel heterogeneous machine is determined by its parallelism and communication behavior.

Because the fixed budget comparison had to be strictly observed and network cost was assumed to track processor cost, slow networks were used with slow processors and fast networks were used with fast processors. Using fast networks in the entire HPAM machine will add to the advantages of HPAM machines over one-level homogeneous machines under the fixed budget criterion. The CCratio can be improved by (1) using fast networks throughout the entire HPAM machine (2) providing hardware and software support for fast collective communication across levels and within levels [21] (3) using computing in memory for the second or third level of an HPAM machine [22].

Finally, as shown by the last two columns of Table 3, in 50% and 80% of the cases for the selected benchmarks, hybrid configurations of the two-level and three-level machines were used, respectively. This indicates that hardware and software support for reconfiguration is needed in HPAM. Furthermore, this support becomes more critical as the number of levels increases.

# References


[1] 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*, 1997, 168-175.

[2] Z. Ben-Miled, A Hierarchical Heterogeneous Solution to High Performance Cost-Efficient Computing. *Ph.D. Thesis,* School of ECE, Purdue University, June 1997.

[3] Z. Ben-Miled and J.A.B. Fortes, A heterogeneous hierarchical solution to cost-efficient high performance computing. *Par. and Dist. Processing Symp.*, 1996, 138–145.

[4] 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.*, 1996, 138–145.

[5] P. Dinda, T. Gross et. al., *The CMU Task Parallel Program Suite*, Tech. Rep. CMU-CS-94-131, Carnegie Mellon Univ., 1994.

[6] W. Blume, R. Doallo et. al., Parallel programming with Polaris. *IEEE Computer*, 1996, 78–82.

[7] 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.*, 1997, 126–136.

[8] J.L. Hennessy and D.A. Patterson, *Computer Architecture A Quantitative Approach*, (San Francisco, Morgan Kaufmann, 1990).

[9] M.L. Barton and G.R. Whiters, Computing performance as a function of the speed, quantity, and cost of the processors. *Supercomputing*, 1989, 759–764.

# 5 Conclusion

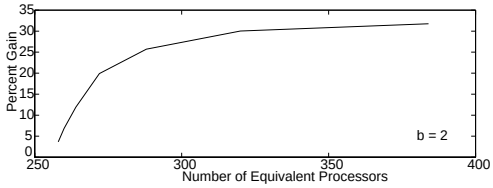
In this simulation-based study each HPAM machine was compared to the optimal one-level homogeneous machine. This optimal machine varies in size and processor speed for a given budget across the benchmarks studied. The results of this study are summarized in Table 3.

| Benchmark | # DoPs | CCratio when # Processors = | | HPAM | Reconfiguration | |
|---|---|---|---|---|---|---|
| | | 2 | 128 | Performance | two-level | three-level |
| Cfft2 | 2 | 225.19 | 196.57 | O L H H | yes | yes |
| Hfft2 | 2 | 24.97 | 1.53 | O O O L | yes | yes |
| Cstereo | 3 | 54.78 | 1.76 | L L L L | no | yes |
| Hstereo | 3 | 11.24 | 0.47 | O O O O | no | yes |
| Cairshed | 5 | 0.87 | 0.41 | O L L L | no | no |
| Hairshed | 2 | 5343.65 | 96.70 | O L H H | yes | yes |

Table 3: Summary of experimental results. The number of distint *DoPs* for each benchmark are listed in the second column. The third and the fourth columns show the CCratio for a homogeneous machine with 2 and 128 processors, respectively. The performance column summarizes the result of the percent gain of HPAM with respect to the optimal homogeneous machine. In this column, the four entries correspond low-, medium-, high- and very high-range budgets (*neq*). These entries can take on one of the following values : O, H, L. The symbol O indicates that a one-level homogeneous machine is the machine of choice for the given combination of benchmark and range of budget. The symbol H indicates that a two-level or a three-level heterogeneous machine is the machine of choice. Furthermore, the ratio of processor speed between levels should be high (i.e. $\geq 8$). Similarly, the symbol L indicates that a two-level or a three-level heterogeneous machine is the machine of choice. However, the ratio of processor speed between levels should be low (i.e. $\leq 4$ ). The last two columns indicate whether reconfiguration was used or not for the given benchmark and machine configuration.

Table 3 indicates that for low budgets, the machine of choice is a homogeneous multiprocessor machine. For medium to high range budgets, four benchmarks have higher performance when executed on a multilevel heterogeneous machine than would have been possible with the optimal homogeneous machine. These benchmarks are Cfft2, Cstereo, Cairshed and Hairshed. For these benchmarks, although a strict fixed budget test was used, percent gains of 10% to 30% with respect to the optimal one-level homogeneous machine can be achieved using a multilevel HPAM machine for mid-range to very high-range budgets. The remaining two benchmarks (i.e. Hfft2 and Hstereo) are best executed on a homogeneous machine for the range of budgets tested.

The communication behavior of a benchmark is as important as its parallelism behavior in determining the performance of the benchmark when mapped onto a heterogeneous machine. As shown in Table 3, Hfft2 and Hstereo are best executed on a homogeneous machine for all ranges of budgets tested. Although these two benchmarks have several degrees of parallelism, they also have a low CCratio that decreases rapidly as the number of processors increases. The CCratio can also dictate the HPAM organization of choice. In Table 3, the organization of choice for Cstereo and Cairshed is a multilevel heterogeneous machine in which the ratio of processor speed between levels is low



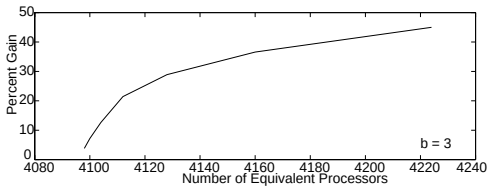
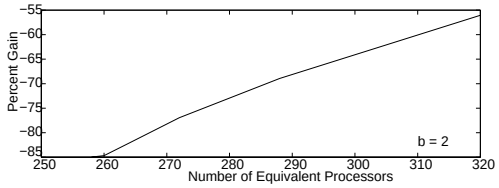




Figure 15: PG of Cfft2 on $H = \{L_{1,16}, L_{1,1+}\}$ when $b = 2$ and $b = 3$.



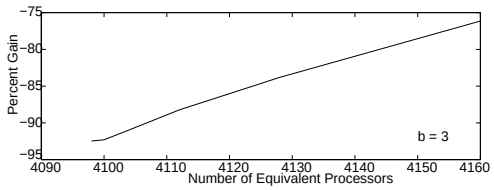




Figure 16: PG of Hfft2 on $H = \{L_{1,16}, L_{1,1+}\}$ when $b = 2$ and $b = 3$.

level homogeneous machine. The result of Figure 13 show, against our expectation, that Hairshed can benefit from a two-level heterogeneous machine.

The difference between the processor speed in the first and second levels was also varied. The result of this experiment indicated that the larger the difference the higher is the percent gain. For example, $H = [L_{8,1}, L_{1,1+}]$ has higher percent gain than $H = [L_{8,1}, L_{2,1+}]$ which in turn has higher percent gain than $H = [L_{8,1}, L_{4,1+}]$. Additionally, for the range of values of $neq$ tested, it was experimentally found that two-level machine organizations with small number of processors in the first level have higher percent gain than those with a larger number of processors in the first level.

Hairshed was also mapped onto three-level machines (Figure 14) with one processor in the first-level and varying number of processors in the second and third level. For this case, a hybrid configuration approach that alternates between the one-level and the two-level configuration of a given machine was used. Figure 14 shows that positive percent gain for the three-level HPAM machines are obtained for values of $neq$ beyond 300.

## 4.4 The Effects of Varying the Cost Model

In this study, the cost of a processor with speed $\alpha$ was assumed to grow in proportion to $\alpha^b$, where $b = 2$. In order to test the sensitivity of the experimental results obtained with respect to $b$, the value of $b$ was varied. Increasing $b$ results in a shift to the right for all the plots and an increase in the difference between the speedup achieved by the optimal homogeneous machines and the heterogeneous machines. This increase translates into an upward shift if the percent gain is positive and a downward shift if the percent gain is negative. Figures 15 and 16 illustrate this behavior for Cfft2 and Hfft2, respectively. In [9], a similar behavior was observed when homogeneous multiprocessor machines using varying processor speed were compared.

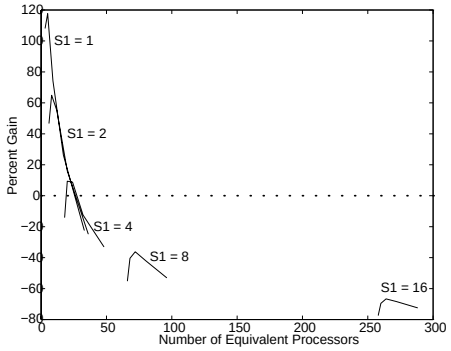



Figure 11: Percent gain of Cairshed on $H = [L_{S1,1}, L_{1,1+}]$.

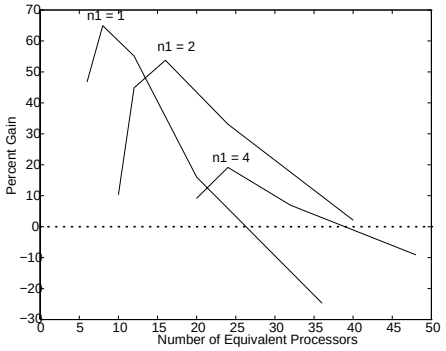



Figure 12: Percent gain of Cairshed on $H = [L_{2,n1}, L_{1,1+}]$.

## 4.3 Airshed Simulation

Figure 11 shows the percent gain when Cairshed is mapped onto two-level machines. This figure indicates that substantial gain can be achieved with an adequate task distribution regardless of speed. However, this increase in gain cannot be sustained as the size of the machine increases. When the number of processors increases in the first level of the two-level machine (Figure 12), the high percent gain can be sustained for a wider range of machine sizes. However, increasing the number of processors in the first level is also associated with a drop in performance. With support for fast reductions in the second level, and the use of one or two fast processors in the first level, it may be possible to achieve high percent gain and be able to sustain it for a wide range of machine sizes.

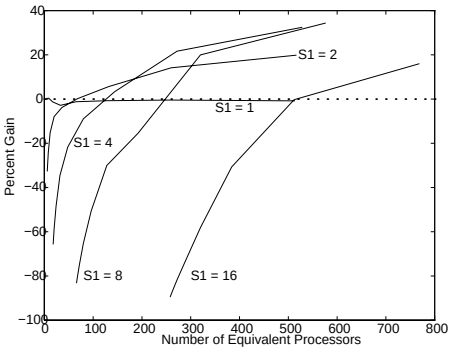



Figure 13: Percent gain of Hairshed on $H = [L_{S1,1}, L_{1,1+}]$.

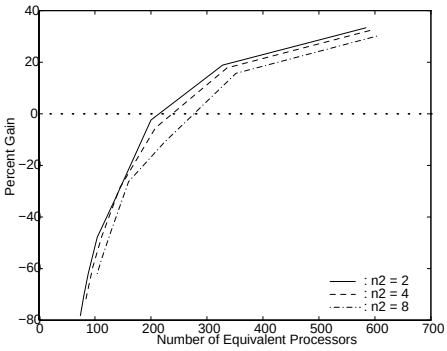



Figure 14: Percent gain of Hairshed on $H = [L_{8,1}, L_{2,n2}, L_{1,1+}]$.

Hairshed was mapped onto a two-level machine using a hybrid configuration approach (i.e., the configuration alternates between the native and the one-level configurations of the two-level machine). Figure 13 shows the percent gain obtained for Hairshed when the processor speed in the first level varies. For any given first level processor speed, the percent gain starts below "zero" and increases toward values above "zero" as $neq$ increases. Unlike the two-level heterogeneous machine organizations, the homogeneous machine $H = [L_{1,1}, L_{1,1+}]$ does not address the effect of Amdahl's law. Thus, as shown in Figure 13, this machine has zero percent gain over the one-

## 4.2 Multibaseline Stereo

Cstereo was mapped onto a two-level machine (Figure 9) by assigning the sequential tasks to the first level and the remaining tasks to the second level. Figure 9 shows that the machine organization $H = [L_{1,1}, L_{1,1+}]$ achieves a peak percent gain of 60% which drops below zero as $neq$ increases. This indicates that Cstereo can benefit from a task distribution that assigns the sequential portion of the application to one processor and distribute the parallel portion among the remaining processors, regardless of whether the machine is homogeneous or heterogeneous.

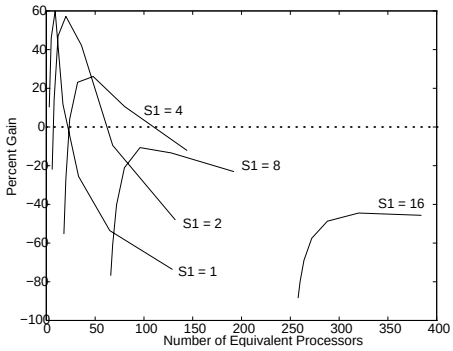



Figure 9: Percent gain of Cstereo on $H = [L_{S1,1}, L_{1,1+}]$.

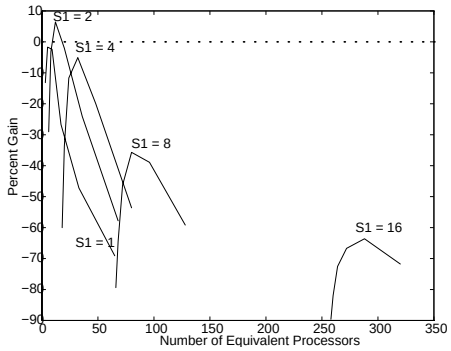



Figure 10: Percent gain of Hstereo on $H = [L_{S1,1}, L_{1,1+}]$.

Mapping Cstereo onto a two-level machine with more than one processor in the first level consisted of manually distributing the sequential portion of the code across the processors in the first level (automatic distribution of this task may not be possible with current parallelizing compiler technology). The results of this mapping showed that ratio of processor speed in the first and the second level should be less than or equal to four (Figure 9) and that Cstereo is best executed on a two-level machine with one processor in the first level.

Cstereo was also mapped onto three-level machines with different organizations. The mapping was such that the second and third levels of the machine were reconfigured into a single level. The result of this mapping indicated that the percent gain of Cstereo decreases as the number of processors in the second level increases. Thus, Cstereo is best executed on a three-level machine with small number of processors in the second level.

Using a mapping similar to the one used for Cstereo, Hstereo was mapped onto two-level and three-level machines. Figure 10 shows the percent gain achieved for Hstereo when the processor speed in the first level varies. This figure indicates that for heterogeneous two-level machines, the percent gain decreases as the processor speed in the first level increases. Additionally, the percent gain for Hstereo is lower than that of Cstereo since Hstereo has lower CCratio than Cstereo. Furthermore, unlike Cstereo, Hstereo has negative percent gain when only task distribution is used (as opposed to both heterogeneity and task distribution). The percent gain of Hstereo when executed on a three-level machine is negative.

machine (percent gain $\leq 10\%$), the machine of choice is a one-level homogeneous multiprocessor machine. As the budget increases, the two-level and three-level machines out-perform the one-level machine. For the range of processor speed tested, the gain reaches 20% and increases as the processor speed in the first level increases. Thus, for mid-range to high-end budgets, Cfft2 is best executed on a two-level or three-level machine with a very fast processor in the first level and slow processors in the remaining levels. Although it was expected that Cfft2 will only map well onto a two-level machine, Figure 6 shows that it is possible for the three-level machine to achieve a percent gain greater than zero if a hybrid configuration is used.

Previous studies ([20] and references there in) have shown that an increase in performance can result from assigning the appropriate number (determined by the characteristics of the task) of identical processors to each task in the application. The mapping of Cfft2 on $H = [L_{1,1}, L_{1,1+}]$ shows that, by itself, assigning the appropriate number of processors to each task might not be sufficient to obtain an increase in performance. In addition to adequate processor-task assignment, the heterogeneous features of HPAM are needed to achieve a gain in performance with respect to a one-level machine.

Several mappings of Hfft2 on a two-level machine were attempted. One of these attempts consisted of mapping the transpose operations in Hfft2 to the first level and usually fastest level of a two-level machine. This mapping resulted in a lower performance than actually mapping Hfft2 using the one-level configuration of the two-level machine. When Hfft2 is executed on a two-level machine the percent gain is negative (Figure 7). The loss in performance is small when the difference in speed between the two levels is also small. As this difference increases, the gap between the speedup of the optimal one-level and the two-level machine increases. This is also indicated by Figure 8 which shows the percent gain for the machines $H = [L_{16,1}, L_{4,1+}]$ and $H = [L_{16,1}, L_{8,1+}]$. Moreover, for a given processor speed, the gap between the speedup of the two machines reduces as $neq$ increases.

Increasing the speed of the processors in the second level with respect to the first processor leads to an improvement in the performance of the two-level machine with respect to the one-level machine. This mainly occurs when the processor speed in the second level is as close as possible to the processor speed of the first level. This indicates that better performance for the two-level machine is achieved when it has a configuration that is closest to the one-level machine. Furthermore, as indicated by the trend of the curves in Figure 8, given a very high budgets ($neq > 4500$) and using two high speed processors in the first and second level (e.g. 64 and 32, respectively). The two-level HPAM machine can outperform the optimal homogeneous machine.

Mapping Hfft2 onto a three-level machine also resulted in a negative percent gain. The negative performance gain for Hfft2 on a two-level and three-level machines is due, among other factors, to the low CCratio of Hfft2.

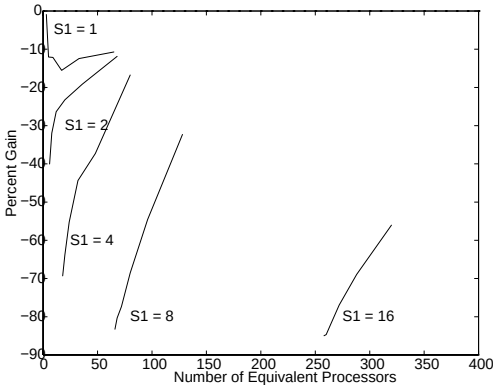



Figure 7: Percent gain of Hfft2 on $H = [L_{S1,1}, L_{1,1+}]$.

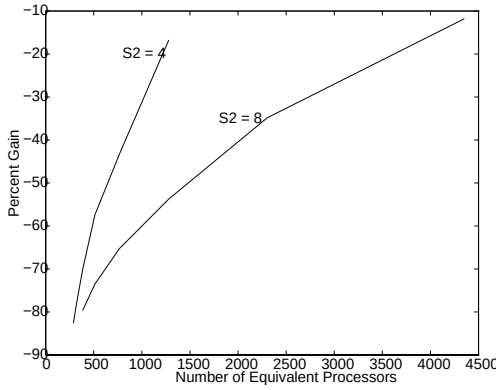



Figure 8: Percent gain of Hfft2 on $H = [L_{16,1}, L_{S2,1+}]$.

applied to the data presented in this section. Additionally, a summary of of the conclusions that can be drawn based on this latter approach are included in Table 3 of Section 5.

Except when the speed in the first level is one, each two-level machine reaches a peak performance above zero in Figure 5. The percent gain at this peak increases as the processor speed in the first level increases with respect to the processor speed of the second level. Additional experiments indicated that the performance degrades as the number of processors in the first level increases.

For a two-level machine, the sequential portion of Cfft2 can be mapped to the first level while the parallel portion can be mapped to the second level. Although this mapping resulted in a percent gain larger than zero, additional gain was obtained by using a hybrid configuration. This hybrid configuration consists of alternating between the native configuration of the machine and a one-level configuration as needed. For example, the mapping of Cfft2 on $\{L_{16,1}, L_{1,1+}\}$ alternates between the native configuration $\{L_{16,1}, L_{1,1+}\}$ and the configuration $\{L_{1,neq(L_{16,1})+1+}\}$.

Figure 6 shows the percent gain obtained by executing Cfft2 on a three-level machine when the processor speed in the second level is two. The behavior of Cfft2 on a three-level machine is similar to its behavior on a two-level machine. The peak performance increases as the speed of the processor in the first level increases. A test was also conducted to evaluate the effect of varying the processor speed in the second level of the three-level machine on the percent gain. This test revealed that as the processor speed in the second level increases, the performance of the machine degrades. The configuration for the three-level machine was also hybrid. However, in this case the native configuration was not used. Only the two-level and one-level configurations of the three-level HPAM machine were used. For both the two-level and three-level machines, the peak performance will continue to increase as the processor speed in the first level increases with respect to the processor speed in the remaining levels.

The above-mentioned results lead to several conclusions about the behavior of Cfft2 and the adequate machine organization for its execution. For machines with a small budget (i.e., $neq \leq 50$), given the low return of a multilevel

homogeneous multiprocessor machine built with processors of speed $\overline{\alpha} = 2$ within a fixed budget ($neq = 100$) has higher speedup than the largest homogeneous multiprocessor machine built with processors of speed $\overline{\alpha} = 1$ and the same budget. The function defined by the collection of the points indicated by "o" is denoted $R(.)$ and is used as a reference. Linear interpolation is used to obtain the values of $R$ corresponding to values of $neq$ that have not been experimentally generated. The number and the speed of the optimal homogeneous machine (with speedup $R$) vary from one benchmark to another. Furthermore, for a given benchmark, these also vary with the budget ($neq$) allocated to the machine.

Let $S_H(x)$ denote the speedup obtained for machine $H$ when $neq = x$. The *percent gain* of $H$ with respect to the optimal one-level homogeneous machine is defined as follows:

$$\text{percent gain}_H(x) = 100\frac{S_H(x) - R(x)}{R(x)} \tag{4}$$

If positive, the percent gain indicates the additional speedup achieved over the optimal one-level homogeneous machine for the same budget.

Figure 5 shows the percent gain of Cfft2 on a two-level machine with one processor of varying speed in the first level as indicated by the value of $S1$. In this figure the "zero" line corresponds to the reference speedup (i.e. $S_H(x) = R(x)$ in Equation 4).

The curve corresponding to $S1 = 1$ in Figure 5 represents the percent gain of a two-level machine, constructed with the same type of processor in the two levels (i.e. homogeneous), with respect to the optimal one-level homogeneous machine. The parallel benchmark used for the case $S1 = 1$ is the same as the one use for the cases $S1 > 1$, whereas the parallel benchmark for the one-level homogeneous machine is different. Thus, comparing the case when $S1 = 1$ to the optimal homogeneous machine allows an accurate assessment of whether the percent gain is due to code transformations or to the heterogeneity of the processors.

As in the one-level homogeneous case (Figure 4), the solid curve in Figure 5 indicated by "o" represents the optimal two-level heterogeneous machine for a given budget. For example, when $300 \leq neq \leq 400$, the two-level machine of choice has one processor of speed 16 in the first level and the maximum number of processors of speed one allowed by the budget (i.e. $neq$) in the second level.

The analysis included in this section (Section 4) compares the speedup of each heterogeneous machine to that of the optimal homogeneous machine (as represented by $R$). This is a strict comparison that favors the homogeneous machine. A more realistic analysis should be based on the comparison of the performance of the optimal heterogeneous machine (e.g. the envelope curve indicated by "o" in Figure 5) and the performance of the optimal homogeneous machine (e.g. the envelope curve $R$, indicated by "o" in Figure 4). This latter analysis approach can be directly

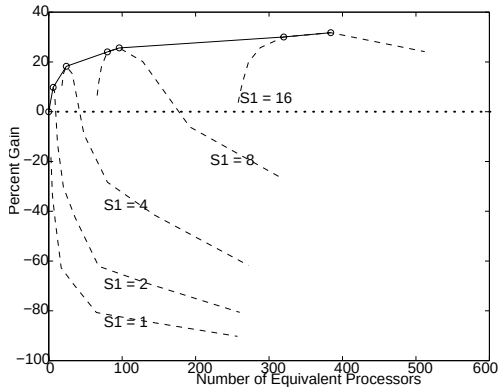



Figure 5: Percent gain of Cfft2 on $H = [L_{S1,1}, L_{1,1+}]$.

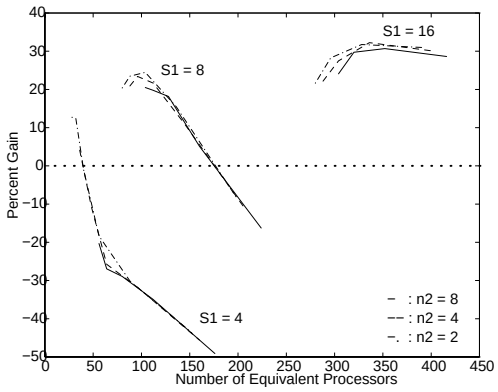



Figure 6: Percent gain of Cfft2 on $H = [L_{S1,1}, L_{2,n2}, L_{1,1+}]$.

network that is balanced with respect to processor speed allows the cumulative cost of the individual processors in the machine to remain indicative of the overall cost of the machine. For a processor with $\overline{\alpha}$ = 100-Mhz, $L$ is equal to 120ns (which does not represent an aggressive network) and for a processor with $\overline{\alpha}$ = 1Ghz, $L$ is equal to 12ns (which represents a more aggressive network). For example, the reported *per-hop-delay* for the Paragon is 40 ns[12].

*Contention*: HPAM_Sim allows the simulation of the target machine network with or without contention. Simulation of network contention is most time consuming. For Hfft2 and Hstereo, there is more than 20% difference between the simulation estimated execution times obtained with and without contention. Thus, for these benchmarks the contention mode of HPAM_Sim was used. For Cfft2, Cstereo, Cairshed and Hairshed, the percent difference between the simulation estimated execution times using the contention mode and the non-contention mode is less than 1%. Thus, for these benchmarks the non-contention mode was used.

# 4 Simulation Results and Analysis

The two versions of each of the selected CMU benchmarks were mapped onto different HPAM configurations. HPAM_Sim was used to simulate their executions. There are two different factors that can determine the assignment of a given task to a given level: the *DoP* and the communication load. For the remainder of this study speedups are computed with respect to the reference processor.

## 4.1 2D Fast Fourier Transform

Cfft2 was executed on a one-level homogeneous multiprocessor. The speedups obtained by varying the number of processors and processor speeds are shown in Figure 4. In this figure, the points indicated by "o" represent the one-level machine with optimal speedup for a given value of *neq*. For the Cfft2 benchmark (Figure 4), the largest

*Communication Latency*: For a given machine, the communication latency ($t_{comm}$) is the sum of three major components[11]: $t_{send}$, $t_{net}$ and $t_{recv}$. For a message of size $S$ Bytes, each of these components can be expressed as follows:

$$t_{send} = C_{send} + L\frac{S}{W}, \quad t_{net} = L_1(\frac{S}{W_1} + n), \quad t_{recv} = C_{recv} + L\frac{S}{W}. \tag{3}$$

In the expression of $t_{send}$ and $t_{recv}$, $C_{send}$ and $C_{recv}$ are constants. In these same expressions, the terms that vary with the size of the message are mainly due to any copy operation required during a send or a receive. The ratio $L/W$ represents the time required to process 1 $Byte$ of data during these copy operations. In the expression of $t_{net}$, $L_1$ is the *per-hop-delay*, $n$ denotes the number of *hops* (between the source and the destination of the message) and $W_1$ is the width of the point-to-point link between a processor and its nearest neighbor. On most commercial multiprocessor systems, the communication delay is dominated by the software startup latency ($t_{send} + t_{recv}$)[12]. For example, in [11], the different parameters in Equation 3 corresponding to the IBM-SP1 were estimated at : $C_{send} = 20\mu s$, $L/W = 0.02\mu s/Byte$, $L_1/W_1 = 0.03\mu s/Byte$ and $C_{recv} = 35\mu s$. The software startup latency is usually estimated as half the time it takes for one processor to send and receive a small message to/from a nearest neighbor ("echo" test).

The variation in software startup latencies can be as high as an order of magnitude for systems using traditional message passing protocols (e.g. Cray-T3D+PVM: $21\mu s$ and SP1+MPL: $270\mu s$ [14]). The focus of several studies such as [15] and [16] is to reduce software startup latency. The common idea behind these approaches is to use the advantages of shared memory communication paradigms in message passing. This is accomplished in [15] by using active messages and in [16] by using virtual mapped memory communication. Both models try to maximize the overlap between computation and communication while performing message passing in user space. In [16], it was shown that one-way transfer times for the "echo" test that are close to the physical hardware latency can be achieved for a message passing communication paradigm. Given the variation in software startup latencies due to different implementations and that new software and hardware techniques will continue to improve the software startup latency, this study focuses on comparing HPAM to homogeneous machines in the ideal case where $C_{send} = C_{recv} = 0$. Other studies will address the effect on the performance of HPAM machines due to varying startup software latencies.

For all the experiments included in this paper, we make the simplifying assumption that $L_1 = L$ and $W_1 = W$. This assumption was motivated by the fact that for the IBM-SP1 (described above) as well as for other machines[12], $L_1/W_1 \approx L/W$. Furthermore, $W$ is set to 8 Bytes because technology advances will allow 4 to 8 Bytes wide channels in the near future [13].

The *per-hop-delay* ($L$) is set to $12/\overline{\alpha}$. This parameter varies with the processor speed in order to reflect the fact that fast networks are used with fast processors and slow networks are used with slow processors. Using a

HPAM machine, the $neq$ is:

$$neq\,(H = [L_{\alpha_1,\beta_1}, \cdots, L_{\alpha_n,\beta_n}]) = \sum_{i=1}^{n} {\alpha_i}^b \beta_i \tag{1}$$

In [9] and [10] a value of $two$ was used for $b$. However, in [9] it was indicated that $b = 2$ is a conservative estimate for the cost of fast processors and that the cost function becomes exponential for high-end processors. The value $b = 2$ is used throughout this study and the effect of varying $b$ is discussed in Section 4.6. For a given value of $b$ and a machine $H$, $neq$ represents the size of the one-level homogeneous machine that can be built using processors with normalized speed one and the same budget allocated for $H$.

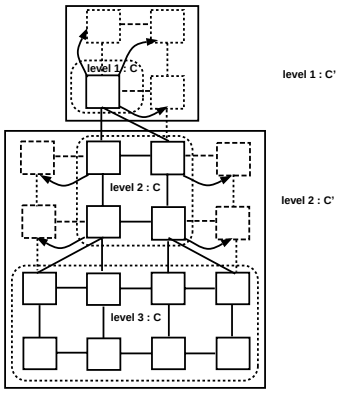



Figure 3: Configurations C' = $\{L_{2,4},\ L_{1,16}\}$ and $C = \{L_{8,1},\ L_{2,4}, L_{1,8}\}$ of $H = [L_{8,1}, L_{2,4}, L_{1,8}]$. The levels of C and C' are indicated by dashed and solid lines, respectively. Processors connected by arrows are emulated by a single physical processor.

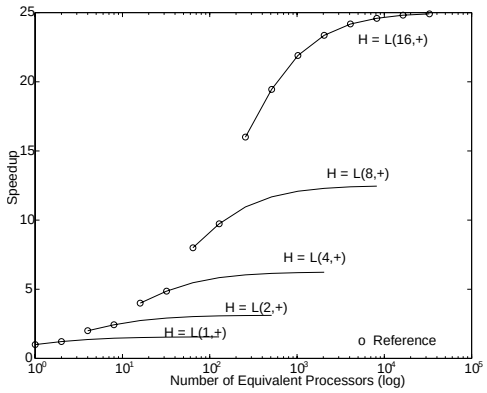



Figure 4: Speedup of Cfft2 on $H = [L_{1+,1+}]$.

## 3.3 Communication Model

*Network Topology*: In concept HPAM can have any interconnection network. However, in this study each level of HPAM uses a mesh for intra-level interconnection network. Let processors be represented by $(row, column)_{level}$ tuples such that the topmost row and the leftmost column of a given level are labeled zero. Also, let level $l$ consist of an $r_l \times c_l$ processor mesh ($r_l$ and $c_l$ are powers of two). Furthermore, let levels be numbered in increasing order starting from the top level. A given processor $(0, a)_l$ is connected to processor $(r_{l-1} - 1, b)_{l-1}$. The relationship between $a$ and $b$ is given by:

$$b = \lfloor a/f_l \rfloor \text{ where } f_l = c_l/c_{l-1},\ \text{and } c_l \leq c_{l-1} \tag{2}$$

The above inter-level network was selected because it allows upper levels to behave as an extension of adjacent lower levels while using existing connectivity efficiently and without the need for additional connectivity.

# 3 Simulation Environment

This section describes the simulation environment used in this study. This environment consists of three major components: machine configuration, cost model and communication model. These components are described in the following subsections.

## 3.1 HPAM Configurations

The *non-normalized* speed of a processor is defined as the ratio of the number of instructions executed per cycle to the cycle time of the processor. The *normalized speed* of a processor refers to the non-normalized speed of the processor divided by the non-normalized speed of the reference processor. The notation $\alpha$ and $\overline{\alpha}$ is used to distinguish between normalized and non-normalized processor speeds, respectively.

Let $L_{\alpha,\beta}$ represent an HPAM level that consists of $\beta$ processors with speed $\alpha$. The notation $L_{\alpha,\beta+}$ is used to represent a family of levels with a number of processors greater than or equal to $\beta$ and processor speed $\alpha$ (e.g. $L_{\alpha,2+}$ represents $L_{\alpha,2}, L_{\alpha,4}, L_{\alpha,8}$ ...). Similarly, $L_{\alpha+,\beta}$ is used to denote the family of levels with $\beta$ processors and processor speed greater than or equal to $\alpha$. Let $H = [L_{\alpha_1,\beta_1}, L_{\alpha_2,\beta_2}, \cdots, L_{\alpha_n,\beta_n}]$ represent an n-level HPAM physical machine. Each HPAM machine can have several virtual HPAM configurations. Let $C = \{L_{\gamma_1,\delta_1}, L_{\gamma_2,\delta_2}, ..., L_{\gamma_m,\delta_m}\}$ represent an m-level configuration, where $m \leq n$. The *native* configuration of the above HPAM machine ($H$) is the configuration $C = \{L_{\alpha_1,\beta_1}, L_{\alpha_2,\beta_2}, ..., L_{\alpha_n,\beta_n}\}$ which matches the physical configuration. Figure 3 shows the native configuration ($C = \{L_{8,1}, L_{2,4}, L_{1,8}\}$) as well as the configuration $C' = \{L_{2,4}, L_{1,16}\}$ of the HPAM machine $H = [L_{8,1}, L_{2,4}, L_{1,8}]$. In $C'$, the second level of $H$ behaves as an extension of the third level and the first level emulates a level with a larger number of slower processors (than the first level of C).

HPAM_Sim was used in this study to simulate the execution of the selected benchmarks on several HPAM machines. Furthermore, when needed, HPAM_Sim was also used to emulate the virtual configurations of a physical HPAM machine.

## 3.2 Cost Model

According to the model proposed in [9], the cost of a given level $L_{\alpha,\beta}$ is estimated as $a\alpha^b\beta$, where $a$ and $b$ are constants. In this study, the reference processor ($L_{1,1}$) has normalized speed *one* and cost $a$. The constant $b$ captures the possibly nonlinear relation between processor cost and speed. The *number of equivalent reference processors* ($neq$) is used to compare different HPAM machines. A fixed value of $neq$ corresponds to a fixed budget. The $neq$ of $L_{\alpha,\beta}$ is the cost of $L_{\alpha,\beta}$ relative to the cost of the reference processor (i.e., $\frac{a\alpha^b\beta}{a} = \alpha^b\beta$). Similarly, for an entire

for stereo will not scale up well as the number of processors increases. This suggests that the machine of choice for stereo should be a two-level organization.

As shown in Table 1, airshed has five distinct $DoPs$ including a small sequential portion. However, the parallel portions of airshed have a high communication overhead (Figure 1). These portions mainly consist of reduction operations. Thus, in order to observe any performance gain by using more than two levels or a two-level organization with large number of processors, the machine would need hardware or software support for fast reductions.

## 2.2 Hand Parallelization

Hand parallelization in this study relied on the detailed analysis included in [5]. Hand parallelization (Table 2) exposes more parallelism with $DoP = 65536 = 256 \times 256$ in fft2. This degree parallelism is due to two nested loops with $DoP = 256$ each. There are two types of tasks with $DoP = 65536$: communication intensive tasks and non-communication intensive tasks. For the hand-parallelized version of fft2, of the 99.99% execution time segment with $DoP = 65536$ in Table 2, 58.33% and 41.66% correspond to the first and second types of tasks, respectively. Due to the communication intensive nature of the first type of tasks, it might not be practical to fully exploit parallelism up to $DoP = 65536$. Thus, a more realistic estimate for the $DoP$ of these tasks is 256 which results from parallelizing one of the two nested loops and serializing the other. Compared to the compiler-parallelized version of this benchmark, the CCratio is lower for the hand-parallelized version (Figure 2). One would expect this benchmark to best map onto a one-level homogeneous machine.

By using scalar-to-array promotion transformations, hand parallelization of stereo exposed more parallelism with $DoP = 61440$. However, this parallelism is associated with high additional computation and control overhead. Given the three distinct $DoPs$ in the hand-parallelized version of stereo, this benchmark is expected to best map onto a three-level HPAM machine. However, due to its low CCratio, a two-level HPAM machine may be more cost-efficient.

Polaris exposed mostly parallelism due to reduction operations in airshed. In the hand-parallelized version, parallelism across the input grid points was detected. This parallelism is hard to detect with a parallelizing compiler because it requires extensive interprocedural analysis. As shown in Figure 2 and Table 2, airshed is mostly parallel and has a high CCratio. This suggests that airshed is best mapped onto a homogeneous one-level organization. The notation Hbenchmark and Cbenchmark (e.g. Hfft2 and Cfft2) refers to the hand-parallelized and compiler-parallelized versions of each benchmark, respectively.

communication not only includes message passing operations but also any copy or setup operations that are needed for communication only. Although this ratio is computed for processor zero only, in most of the benchmarks it is a relatively accurate estimate of the computation and communication times distribution. This ratio depends on the time it takes to transfer data between processor relative to the processor speed. The CCratio also depends on the underlying communication model which is discussed in Section 3.

| DoP | fft2 | stereo | airshed |
|---|---|---|---|
| 1 | 63.6% | 16.8% | 5.4% |
| 25 | | | 22.2% |
| 150 | | | 3.3% |
| 256 | | 18.0% | |
| 400 | | | 61.2% |
| 1000 | | | 7.9% |
| 61704 | | 65.2% | |
| 65536 | 36.4% | | |

Table 1: Parallelism distribution for the compiler-parallelized CMU benchmarks.

| DoP | fft2 | stereo | airshed |
|---|---|---|---|
| 1 | 0.01% | 11.63% | 0.14% |
| 40 | | 28.72% | |
| 700 | | | 99.86% |
| 61440 | | 88.37% | |
| 65536 | 99.99% | | |

Table 2: Parallelism distribution for the hand-parallelized CMU benchmarks.

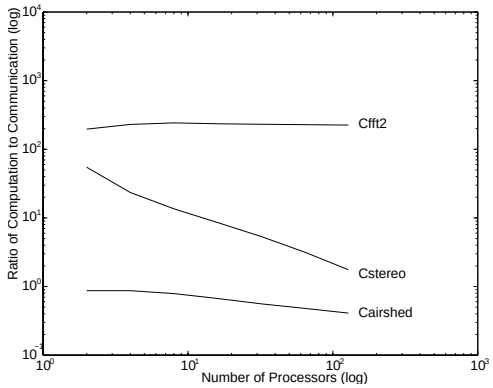



Figure 1: Ratio of the total time spent in computation to the total time spent in communication for the compiler-parallelized benchmarks.

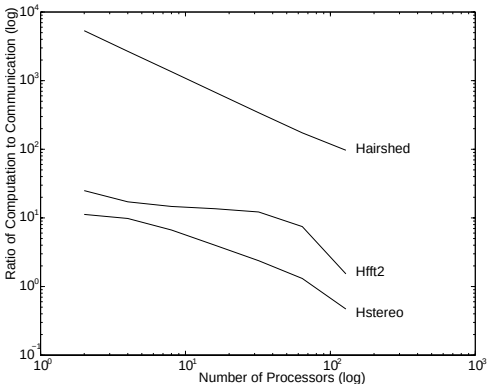



Figure 2: Ratio of the total time spent in computation to the total time spent in communication for the hand-parallelized benchmarks.

Table 1 and Figure 1 show that the CMU benchmarks have different parallelism and communication distributions. The 2D Fast Fourier Transform (fft2) benchmark has two predominant $DoPs$ : a large sequential portion (i.e., $DoP = 1$) and the remaining portion of the application is highly parallel (i.e., $DoP = 65536$). Expectedly, fft2 should benefit from a two-level organization with one fast processor in the first level and many processors in the second level.

The Multibaseline Stereo (stereo) has three distinct $DoPs$. However, for machines with 100 to 200 processors the difference between exploiting parallelism with $DoP = 256$ and $DoP = 61704$ cannot be observed. Furthermore, Figure 1 shows that the CCratio of stereo decreases rapidly as the number of processors increases. Thus, the speedup

each additional level has an increasing number of slower processors. This study addresses three important questions: (1) Given a fixed budget can an HPAM organization compete with the optimal one-level homogeneous machine for a given benchmark? (2) What characteristics of the benchmarks dictate how well a given application maps onto an HPAM organization? (3) How critical is it to provide hardware and software support for reconfiguring a physical HPAM machine with $k$ of levels into a virtual machine with $j < k$ levels (i.e., making one level behave as an extension of its successor level)?

Section 2 of this paper describes general characteristics of the CMU benchmarks[5] used to carry this study. Due to space limitations, only the results corresponding to three out of the five CMU benchmarks are included in this paper. Additional findings about the selected benchmarks as well as the remaining two benchmarks from the CMU suite can be found in [1, 2]. The simulation environment and the notation used are discussed in Section 3. The results obtained by mapping the CMU benchmarks onto different HPAM organizations are included in Section 4. Conclusions are provided in Section 5.

# 2 Benchmark Characteristics

Parallelism in each benchmark is detected by two different approaches: a parallelizing compiler (Polaris[6]) and "hand" parallelization. The first version of these benchmarks is representative of HPAM codes that might result from a parallelizing compilation. The benchmarks that result from the second approach allow the study of performance of HPAM architectures when used by expert programmers.

## 2.1 Compiler Parallelization

Let $DoP$ (degree of parallelism), at a given point in time in the execution of a benchmark, denote the maximum number of assembly instructions that can be executed concurrently according to the parallelism explicilty expressed in the parallelized benchmark. Data parallelism in each benchmark was detected using the parallelizing compiler Polaris. A heterogeneous multiprocessor simulator, HPAM_Sim[7], was then used to evaluate the percentage of time (measured in cycles) during which a benchmark executed with each distinct $DoP$. Table 1 shows the $DoP$ time percentages for the selected CMU benchmarks (percentages under 1% are omitted). The information included in Table 1 does not reflect the communication behavior of the benchmarks. In order to gain insight into the communication behavior of the benchmarks, the ratio of the total time spent in computation to the total time spent in communication was estimated for each benchmark (Figure 1). This ratio (denoted CCratio) is obtained by tracing the communication and computation tasks for one processor (i.e., processor zero) of a one-level organization. The total time spent in

# On the Cost-efficiency of Hierarchical Heterogeneous Machines for Compiler- and Hand-Parallelized Applications*

Zina Ben Miled
Department of EE
Purdue University
Indianapolis, IN46202

José A. B. Fortes
School of ECE
Purdue University
West Lafayette, IN 47907-1285

Rudolf Eigenmann
Department of ECE
Northwestern University
Evanston, IL 60208-3118

Valerie Taylor

**Abstract**

This paper presents a simulation-based analysis of the performance of compiler- and hand-parallelized versions of the CMU benchmarks on heterogeneous machines organized as a hierarchy of processors-and-memory (HPAM). The characteristics of these benchmarks are first established and then each benchmark is mapped onto different multilevel heterogeneous organizations. The performance of these mappings is compared to that achieved by executing the benchmarks on the *optimal* one-level homogeneous machine organization for a given benchmark and machine budget. This study establishes that heterogeneous machine can have higher cost-efficiency than the optimal homogeneous machine. Experimental results show that the heterogeneous multilevel machine organization can benefit from hardware and software support for dynamically reconfiguring three-level and two-level machines into two-level and one-level organizations. Furthermore, these experimental results indicate that the performance of a given application, when executed on a multilevel heterogeneous organization, depends not only on the parallelism distribution but also on the ratio of total communication time to total computation time.

**Keywords**: Heterogeneous, Cost-efficiency, Reconfiguration, Simulation, Parallel and Performance.

# 1 Introduction

The potential for heterogeneous architectures to provide cost-efficient high performance computing has been shown analytically in several previous studies ([3], [4] and references therein). This paper reports the results of a simulation-based study of heterogeneous machines organized as a hierarchy of processors-and-memory (HPAM) [3]. An HPAM consists of several levels of processors-and-memory (PAM) systems. Relative to the first (top) level of the hierarchy,

*This research was supported in part by NSF grants ASC-9612133, ASC-9612023 and CDA-9617372.