A Comparison of Multi-Core Processors on Scientific Computing Tasks

Ali Khajeh-Saeed
Department of Mechanical and Industrial Engineering
University of Massachusetts, Amherst, MA, 01003, USA
khajehsaeed@ecs.umass.edu

Stephen Poole
Computer Science and Mathematics Division, Oak Ridge National Laboratory,Oak Ridge, TN 37831, USA
spoole@ornl.gov

J. Blair Perot*
Theoretical and Computational Fluid Dynamics Laboratory
University of Massachusetts, Amherst, MA, 01003, USA
perot@ecs.umass.edu

ABSTRACT

Many core processors are now becoming widely commercially available. The possibility of using these types of processors for high performance computing applications is therefore of some interest. This work will compare the performance and power consumption of the Tilera Pro64 processor, with a quad-core CPU and with a GPU on three different scientific computing benchmarks: GUPS (memory access speed), Large Sparse Matrix Multiply (scientific computing), and Smith-Waterman sequencing (bio-informatics). The performance of the Tilera, CPU and GPU will be investigated as a function of the number of cores used and as a function of the energy consumption to complete a task.

Keywords

Many-Core, Tilera, Giga Update per Second, Sparse Vector-Matrix Multiplications, Smith-Waterman Algorithm

1. INTRODUCTION

Processors with a large number of on chip cores are now becoming commonly commercially available. Tilera processors have been available since late 2007. They currently come with 64 cores, and 100 cores will be shipped in 2012. In November 2011, AMD announced its 16-core processor. Intel released a 48-core prototype to University researchers in April 2011 and will release Knight's core late 2012 to 2013 for high performance computing applications. One important aspect of these many-core designs is their low power consumption per core. The Intel chip reportedly draws no more than 125 W for all 48 cores. The Tilera-64 draws only about 50 W for 64 cores. Since power consumption, and the resultant cooling costs, can dominate supercomputer costs it is some interest to evaluate the potential of these many core chips for typical supercomputing applications. Our evaluation of these many-core architectures will focus on scalability and performance per Watt when applied to scientific computing applications.

The Tilera and Intel marketing plans are focused on commercial server farms, not supercomputers for scientific computing. However, high performance computing (HPC) is one potential application for these processors. GPUs provide similar computing benefits and are also now being incorporated into recent supercomputer designs. This project will evaluate the potential performance attributes of many-core processors on a few select and hopefully reasonably representative scientific computing benchmarks.

In 2007, Tilera launched its first many-core architecture with 64-core on a single chip. The main goals in Tilera architecture are to provide high performance cores that communicating via cache-coherent iMesh interconnect network architecture and low power hardware [19]. Figure 1 shows Tile processor hardware architecture with detail of an individual tile's structure.

The iMesh interconnect architecture provides high bandwidth and low latency communication between tiles. Each tile is a powerful, full-featured computing system that can independently run an entire symmetric multi-processing operating system. Each tile operates at 866 MHz and implements a 32-bit integer processor engine utilizing a three-way Very Long Instruction Word (VLIW) architecture with 64-bit instruction bundle. Each tile has 16K L1 instruction cache, 8K L1 data cache and unified 64K L2 cache. Also there is 4MByte L3 cache distributed across tiles. There are four on-chip-memory controllers, each supporting 64-bit DDR2 DRAM (with optional ECC protection) and operating at 6.4 GB/s, for a peak memory bandwidth of 25.6 GB/s. Tilera also has two full-duplex XAUI-based 10Gb Ethernet, two 4-lane PCI Express ports and two onboard 10/100/1000 Ethernet MACs with RGMII (Reduced Gigabit Media Independent Interface) interfaces [20].

There are number of previous benchmark implementations on the Tilera [2, 3, 4, 7, 8, 10, 12, 13, 17, 21, 22, 23].

Bornstein et al. [4] implemented image analysis onboard a Mars rover. But Tilera processors don't support floating point form hardware so all floating point operations are emulated in software that causes reduced performance for Tilera. They reported that a conversion from floating point input data to integer increased the speed by a factor of 5. When using many cores they achieved 8 and 9.7 times the

*Corresponding Author

InPar '2012 San Jose, CA, USA

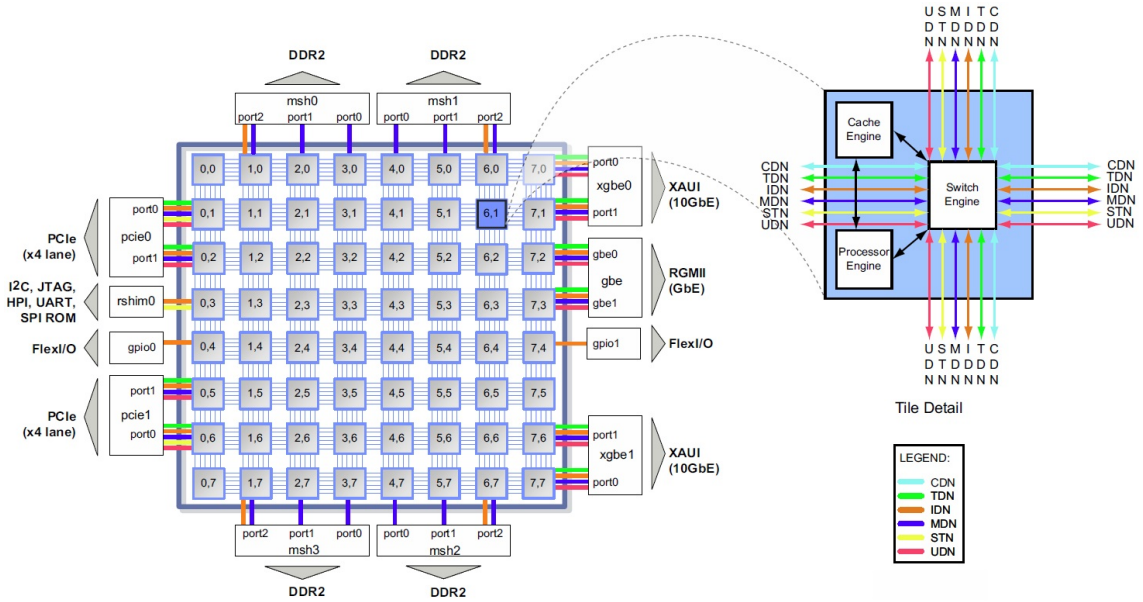


Figure 1: Tile processor hardware architecture with detail of an individual tile's structure (Figure from Tilera data sheet [20])

speed of a single tile (single core) when using 16 and 32 tiles respectively.

Ha et al. [10] studied the scalability problem for dynamic analysis on TILE64 processor. They obtained a benefit from using the fast inter-core communication between tiles for dynamic analysis.

Berezecki et al. [3] implemented key-value store Memcached on TILEPro64 and compared results with a 4-core Intel Xeon L5520 and an 8-core AMD Opteron 6128 HE. They achieved 5 times speedup compared to a single core of CPUs and almost 2 times speed up compared to all the cores of CPUs. They also reported that the TILEPro had 3 to 4 times better performance/Watt compared to the CPUs.

Yan et al. [23] implemented the accelerated deblocking filter of the H.264/AVC. They achieved an overall decoding speedup of 1.5 and 2 times for the HD and SD videos.

Richardson et al. [17] implemented some space applications on TILEPro64 processor. They achieved 23 times speedup compared to the single tile when 32 tiles were used for the sum of the absolute difference. They also reported linear speedup up to 8 tiles.

Ulmer et al. [21] applied a text document similarity benchmark based on the Term Frequency Inverse Document metric on the Tilera and an FPGA and compared results with sequential CPU runs. They achieved 4 times speedup for the Tilera when compared to the single core of 2.2 GHz x86 processor.

In this chapter, we are interested in HPC benchmarks related to scientific computations and the scalability, performance, and power consumption of the Tilera processor. The benchmark cases are: GUPS, large unstructured sparse matrix multiply the Smith-Waterman algorithm (SSCA#1 kernels 1) and 3D FFT.

2. GIGA UPDATE PER SECONDS (GUPS)

The GUPS benchmark [1] was ported to the Tilera. The main goal in this benchmark is to test the random memory access speed of the hardware. The CPU and GPU version of the GUPS benchmark were also used in order to compare the Tilera results with other known architectures. The computations were iterated 10 times in order to get accurate timings. The results presented herein are based on the average of the 10 iterations.

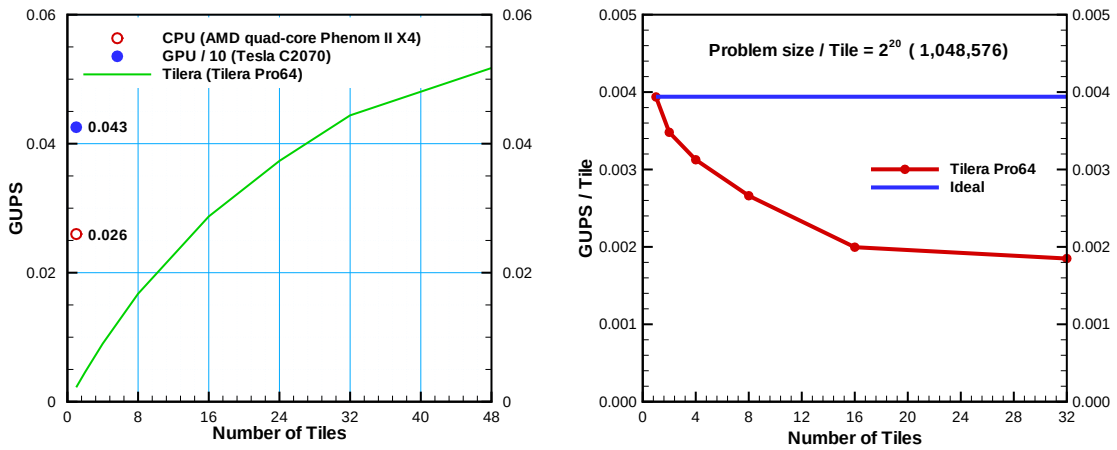
2.1 Strong and Weak Scaling

Figure 2 shows the GUPS (giga-updates per second) for the strong scaling (a constant problem size with varying numbers of tiles) and the GUPS/Tile for weak scaling (problem size per tile is kept constant at 1M) cases. This figure shows that the GUPS is linear with the number of cores only up to 4 cores (because there are 4 memory channels on the Tilera). With larger numbers of cores, the GUPS is below the linear ideal performance. With 48 cores active (out of a maximum of 63) the Tilera is roughly 2 times faster than one core of the CPU and 8 times slower than GPU for strong scaling(the GPU performance is divided by 10 to more easily place it on the graph). For the weak scaling, as the number of tiles increases the GUPS/Tile decreases. This occurs because the tiles must share the limited memory bandwidth.

Because the Tilera code is parallel, there is a possibility of a read before write conflict. The Tilera's results were compared with the CPU results (which do not have this issue). The error rate was negligible and a low level of error is said to be acceptable in the GUPS benchmark specification.

2.2 Power Consumption

Table 1 and figure 3 show the results for power consumption for the GUPS benchmark. The Tilera Pro64 uses 9 times less energy compared to a single core of the AMD quad-core Phenom II X4. But compared to the GPU, the energy consumption is roughly equal. The energy efficiency is given by,



(a) (b)

Figure 2: GUPS for the (a) strong and (b) weak scaling cases for GUPS benchmark compared to a single core of an AMD quad-core Phenom II X4 (red circle) and compared to a Tesla C2070 GPU (blue circle)

Table 1: Strong scaling results for GUPS benchmark for Tilera Pro64 comparing the power consumption to the single core of an AMD quad-core Phenom II X4 and Tesla C2070 GPU

CPU Power (W)	GPU Power (W)	# of Tiles	Power (W)	Efficiency (CPU)	Efficiency (GPU)
69	120	1	2	3.1	0.3
		2	3	4.2	0.4
		4	4	6.2	0.6
		8	5	9.2	0.9
		16	10	7.9	0.8
		24	13	7.6	0.8
		32	15	8.1	0.8
		48	20	6.7	0.7

$$EnergyEfficiency = \frac{Power_{(CPUorGPU)} \times Time_{(CPUorGPU)}}{Power_{(Tilera)} \times Time_{(Tilera)}} \quad (1)$$

A low efficiency is better as it implies less energy is used to complete a task. An efficiency less than one implies the CPU or GPU is more energy efficient than the Tilera.

Table 1 shows that for this benchmark, power consumption for tilera (with 48 tiles), CPU (single core) and GPU are 20, 69 and 120 Watts, repectively.

3. SPARSE VECTOR-MATRIX MULTIPLICATION

Sparse vector-matrix multiplication is a common random memory operation found in many high performance computing codes. The equation below shows the core formula for

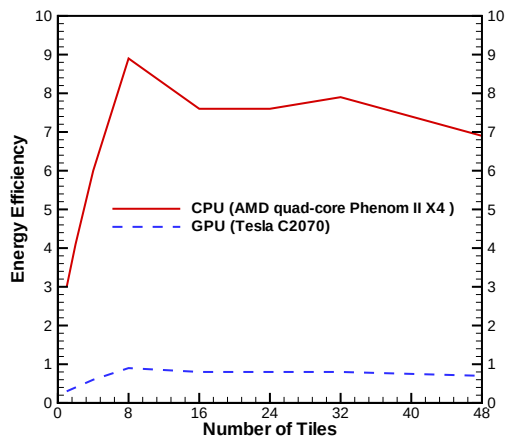


Figure 3: Power efficiency for GUPS benchmark compared to a single core of an AMD quad-core Phenom II X4 (red line) and compared to a Tesla C2070 GPU (blue line).

the 7-point-stencil sparse vector-matrix multiplication that represents the classic discrete 3D Laplacian operation.

$$D_{i,j,k} = B_{i,j,k} \times \begin{pmatrix} (A_{i+1,j,k} + A_{i-1,j,k}) \times dxic_i \times dxiv_i + \\ (A_{i,j+1,k} + A_{i,j-1,k}) \times dyic_j \times dyiv_j + \\ (A_{i,j,k+1} + A_{i,j,k-1}) \times dzic_k \times dziv_k \end{pmatrix} + A_{i,j,k} \times C_{i,j,k} \quad (2)$$

Two different algorithmic approaches on the Tilera were tested: plane marching and block marching. In the plane marching, every tile is responsible for a XY plane from the 3D domain. In the block marching approach, the 3D domain is divided into hexahedra subdomains of size $8 \times 8 \times NZ$, and every tile is responsible for computing a subdomain. This is similar to the GPU layout (which uses $16 \times 16 \times NZ$ subdomain for each GPU multiprocessor). Plane and block marching have different advantages. In the plane marching approach hardware can load the XY plane to its own cache (small problem size). In the block marching approach, hardware can load $z-1$ and z data items into its cache. We tested different block sizes and 8×8 is the most efficient block size for this approach on the Tilera Pro64 card.

3.1 Performance Results

Figure 4 shows the MCUPS (millions of cell updates) and speedup for $128 \times 128 \times 128$ and $256 \times 256 \times 256$ total problem sizes.

In order to show GPU result on the same graph, the MCUPS for the GPU is divided by 100. MCUPS are millions of cell updates (or results, D_{ijk}) computed per second. Each result formally requires reading 7 data values (A) and two constants (B and C) and performing a number of additions and multiplications.

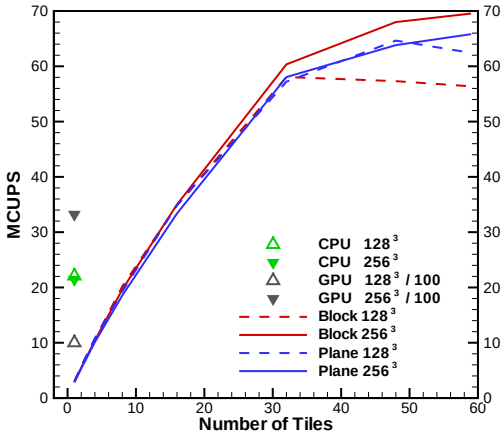
For both algorithm approaches on the Tilera the performance is very similar. For small sizes, plane marching has slightly better performance than block marching. On the other hand, block marching has better performance than plane marching for large problems. A speedup of around $3\times$ is obtained for sparse vector-matrix multiplication on very large matrices 256^3 when the whole Tilera is compared to 1 core of the AMD CPU.

3.2 Power Consumption

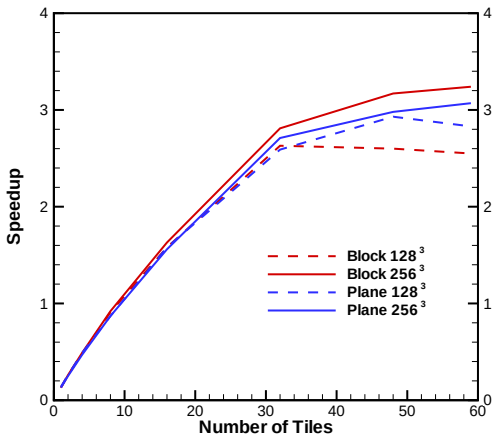
Table 2 shows the power consumption results for Sparse Vector-Matrix multiplication benchmark. The Tilera Pro64 uses up to 21 times less power than a single core of AMD quad-core Phenom II X4. However, it is perhaps better to look at the energy efficiency of the Tilera, which is the total energy used vs. the energy used by the CPU or the GPU.

The table 2 shows that the Tilera is $16\times$ more energy efficient than the CPU and $5\times$ less efficient than the GPU. However, in practice, the Tilera requires a host (which draws 200-300 W of base power). If we were to account for the base power and use 4 (rather than 1 core) of the CPU, the Tilera would have the same efficiency as the CPU.

Figure 5 shows the speedup and energy efficiency of the Tilera Pro64 compared to a CPU, GPU. The results show that up to 8 tiles the energy efficiency increases, and after that with increasing the number of tiles the energy efficiency is the same or decreasing. This shows that the power consumption on the Tilera is directly proportional to the number of memory accesses being performed.

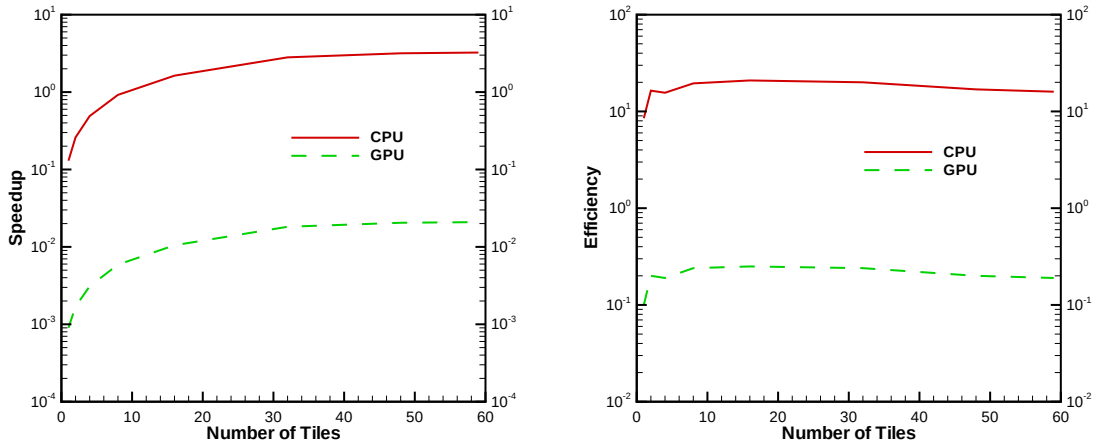


(a)



(b)

Figure 4: (a) MCUPS for 128^3 and 256^3 problem sizes using a Tilera Pro64 (with different numbers of tiles), a single core of an AMD quad-core Phenom II X4, and a Tesla C2070 GPU. (b) Speedup of the Tilera versus one core of the AMD CPU



(a) (b)

Figure 5: (a) Speedup and (b) energy efficiency for the 256^3 problem size for Tilera Pro64 compared to the single core of an AMD quad-core Phenom II X4 and compared to a Tesla C2070 GPU.

Table 2: Power consumption for sparse vector-matrix multiplication ($256 \times 256 \times 256$) for Tilera Pro64 comparing to the single core of AMD quad-core Phenom II X4 and Tesla C2070

CPU	GPU	Tilera Pro64			
Power (W)	Power (W)	# of Tiles	Power (W)	Energy Efficiency (CPU)	Energy Efficiency (GPU)
64	120	1	1	8.5	0.10
		2	1	16.4	0.20
		4	2	15.6	0.19
		8	3	19.5	0.24
		16	5	20.9	0.25
		32	9	20.0	0.24
		48	12	16.9	0.20
		59	13	16.0	0.19

4. SMITH-WATERMAN ALGORITHM

4.1 Anti-Diagonal Algorithm

Two different approaches to solving the Smith-Waterman algorithm were developed for the Tilera Pro64. The first one is the anti-diagonal approach. Because every number in the Smith-Waterman table anti-diagonal is independent, it is possible to do the calculation in parallel. Table 3 shows the results for this approach. The same approach was used on the CPU to get a fair comparison.

With this approach the MCUPS (millions of cell updates per second) is less than or equal to a single core of the AMD quad-core Phenom II X4. The problem with this approach is, when marching along the diagonal, the memory accesses miss the cache and the Smith-Waterman algorithm is then limited by the random memory access speed.

Table 3: Results for anti-diagonal Smith-Waterman algorithm with 59 tiles compared to a single core of an AMD quad-core Phenom II X4

Size	CPU		Tilera Pro64 (59 Tiles)		
	Time (s)	MCUPS	Time (s)	MCUPS	Speedup
1024 x 1024	0.261	4.025	2.236	0.469	0.117
2048 x 2048	1.250	3.356	4.319	0.971	0.289
4096 x 4096	5.155	3.255	9.454	1.775	0.545
8192 x 8192	23.478	2.858	19.518	3.438	1.203

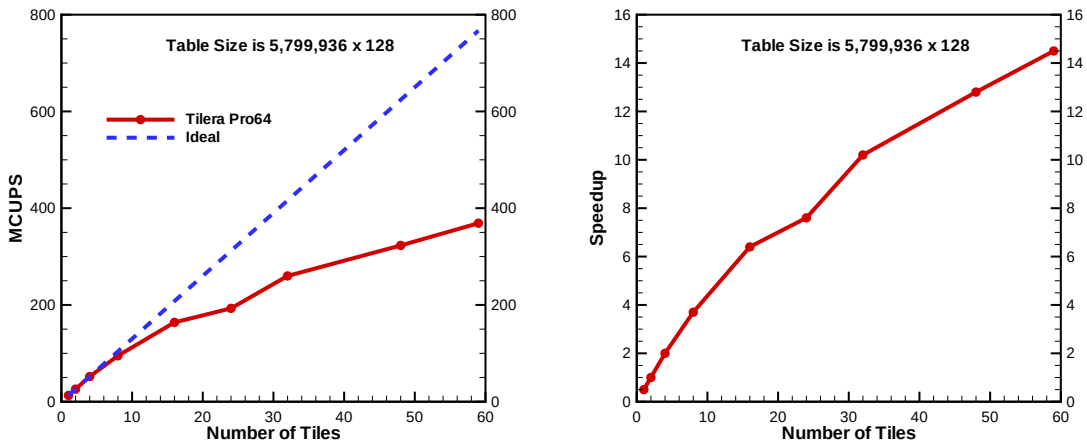
4.2 Row Approach

In the second approach (row approach) the database is divided between the tiles and each subsection of the Smith-Waterman table is calculated in parallel [15, 14]. In this approach some table overlap is necessary to overcome any dependencies between the subsections of the table. For large Smith-Waterman problems the overlap length is small compared to the length of each subsection and is negligible.

4.2.1 Strong Scaling

Figures 6 shows the result for strong scaling (problem size is constant) for kernel 1 of the SSCA#1 benchmark. In this kernel, the Smith-Waterman table is computed and the largest table values (200 of them) and their locations in the table are saved. These are the end points of well aligned sequences, but the sequences themselves are not constructed or saved.

Almost 15× speedup is obtained for the table evaluation compared to a single core of an AMD quad-core Phenom II X4. Note that the MCUPS for the row-access method on the Tilera is almost 100× faster than anti-diagonal approach.



(a) (b)

Figure 6: Strong scaling for kernel 1 (a) MCUPS and (b) speedup for row-access Smith-Waterman algorithm with Tilera Pro64 compared with a single core of an AMD quad-core Phenom II X4.

Table 4: Weak scaling results for row approach Smith-Waterman algorithm for kernel 1 with single core of AMD quad-core Phenom II X4 and Tilera

Size	CPU		Tilera Pro64		
	Time (s)	MCUPS	Time (s)	MCUPS	Speedup
32768	0.13	32	0.16	26	0.8
65536	0.26	32	0.16	52	1.6
131072	0.41	41	0.17	96	2.4
262144	1.33	25	0.21	163	6.3
524288	2.67	25	0.33	201	8.1
786432	3.97	25	0.43	232	9.2
1048576	5.28	25	0.47	284	11.2
1572864	7.92	25	0.62	323	12.8
1933312	9.86	25	0.71	348	13.9

4.2.2 *Weak Scaling*

Figures 7 shows the results for weak scaling (problem size per tile is constant) for kernel 1. The Tilera gets almost 14× speedup, compared to a single core of an AMD quad-core Phenom II X4.

Table 4 shows the MCUPS and speedup for weak scaling with row approach for Smith-Waterman algorithm. This uses 32,768 table entries per tile.

4.3 Power Consumption

Table 5 shows the power and energy consumption results for the first kernel of the sequence matching benchmark (SSCA#1). The Tilera Pro64 uses from 30-50 times less energy compared to the single core of an AMD quad-core Phenom II X4. It was determined that the problem size doesn't have any effect on power consumption.

Table 5: Results for first kernel of SSCA#1 for Tilera Pro64 comparing to the single core of AMD quad-core Phenom II X4

5799936 x 128	CPU		Tilera Pro64			
# of Tiles	Time (s)	Power (W)	Time (s)	Speedup	Power (W)	Energy Efficiency
1			58.68	0.5	1	29.9
2			28.65	1.0	2	30.6
4			14.34	2.0	4	30.6
8			7.81	3.7	5	44.9
16	29.21	60	4.53	6.4	8	48.4
24			3.83	7.6	10	45.8
32			2.86	10.2	12	51.1
48			2.29	12.8	18	42.5
59			2.01	14.5	22	39.6

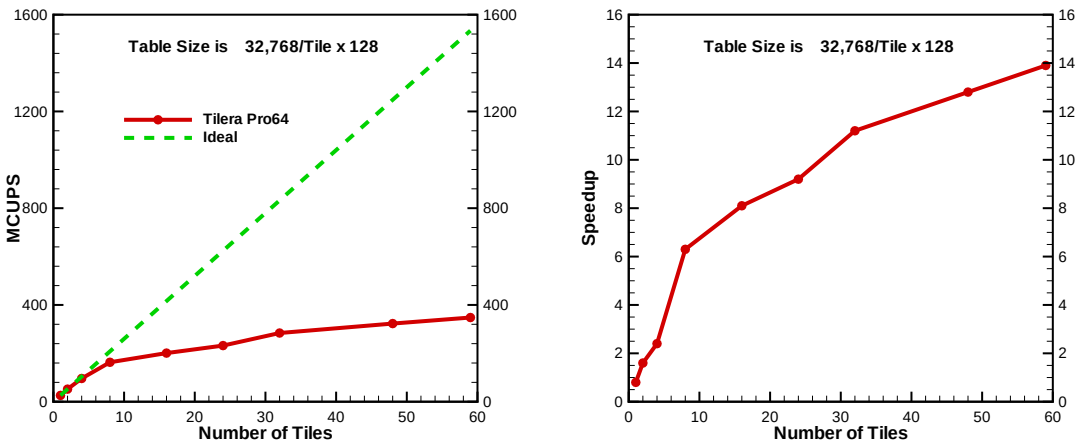
Figure 8 shows the power and energy efficiency of the Tilera Pro64 compared to a single core of an AMD quad-core Phenom II X4. The energy efficiency is given by Equ. **??**.

Note that after 8 tiles the energy consumption is constant, so the power draw is directly proportional to the work being performed.

It should also be noted that these energy efficiency calculations do not include the base power. In practice the Tilera and the AMD quad core require a host that draws 200-300 W when idle. This base power (of the host) will dominate the energy consumption. For this reason, the additional power the Tilera or the CPU draws when operating is essentially negligible in practice.

5. CONCLUSIONS

We have implemented three important high performance computing benchmarks on the Tilera Pro64. Like every multi and many core architectures, the Tilera has some ad-



(a) (b)

Figure 7: Weak scaling for kernel 1 (a) MCUPS and (b) speedup for row-access Smith-Waterman algorithm with Tilera Pro64 compared with a single core of an AMD quad-core Phenom II X4.

vantages and disadvantages. We have divided these features into seven different design metrics; performance, bandwidth, price, software, hardware power and scalability.

Based purely on speed, the Tilera is roughly competitive with a CPU and significantly slower than a GPU. The reason for the relatively low speeds is primarily due to the design of the memory subsystem.

Memory bandwidth is the most critical hardware performance measure for all the benchmarks tested in this work. The Tilera Pro64 has less bandwidth to main memory than a typical CPU or GPU. For example the Tilera Pro64 has maximum of 14 GB/s and modern CPUs are around 40 GB/s and GPUs are roughly 180 GB/s. Newer GPUs have an L1 and an L2 cache (like the CPU and Tilera). The newest Tilera (100-core, not yet released) uses DDR3 instead of DDR2 so its maximum theoretical bandwidth is close to a modern CPU (around 65 GB/s). GPUs currently use DDR5 memory. The Tilera has essentially 4 memory channels to main memory. After 8 tiles start accessing memory, the performance with more tiles is relatively small. It has a fast inter-tile communication network, but we found it very difficult to use this hardware characteristic to any substantial advantage for the 4 benchmarks presented here.

One can buy the latest GPU for around $500-$3000, an Intel CPU (10/20 core/thread E7 Xenon series) for around $4000, and an AMD 12-core for around $1600. The new Tilera with 100 tiles (cores) costs around $11000. It seems that Tilera hardware is expensive compared to CPUs and GPUs. This is probably due to the economies of scale and the high demand for CPU and GPU products outside the area of high performance computing.

The CPU and the GPU (CUDA and OpenCL) programming languages are well known and more up to date than the Tilera's Multicore Development Environment (MDE). However, we should mention that changing codes that are already written in C (not C++) to the Tilera language is very straightforward. One just needs to modify the algorithm in order to take advantage of the Tilera architecture.

The Tilera appears to be compatible with only a few system configurations. For example, the available MDE compilers are compatible with only three specific versions of Linux and three hardware systems (CPU and motherboards).

The Tilera uses 75% less power (above the base/host power) Roughly 24 W extra for the whole TilePro64. Whereas a GPU takes roughly 115 W per GPU and a 4-core CPU takes about 110 W for all four cores. More power means more heat and more cooling. So saving 1 W in power also means saving another watt in cooling costs. Note however, that the Tilera requires a host computer a typical host takes 200-300 W to run so that the power savings of the Tilera are actually highly marginalized by the power costs of maintaining the Tilera's host. In addition, the fast speed of the GPU means that it typically consumes less total energy for a task than the Tilera even though its power consumption is much higher.

Only one Tilera card was tested in this work. However, the Tilera has a number of interconnect options on the card itself. So unlike a GPU, it may be possible to bypass the CPU host and directly connect the Tilera cards together. This might alleviate the MPI bottleneck currently experienced when using many GPUs in a cluster environment.

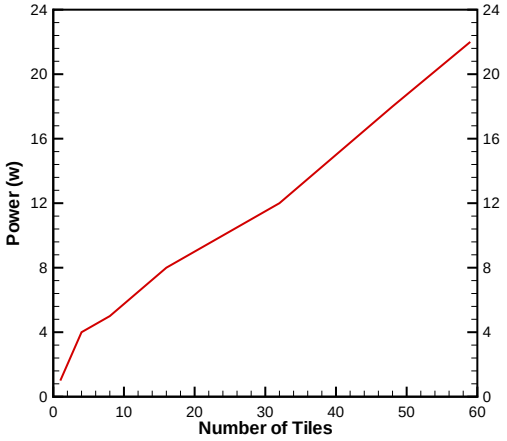
6. ACKNOWLEDGMENTS

The project was supported primarily by the Dept. of Defense via a subcontract from the Oak Ridge National Laboratory.

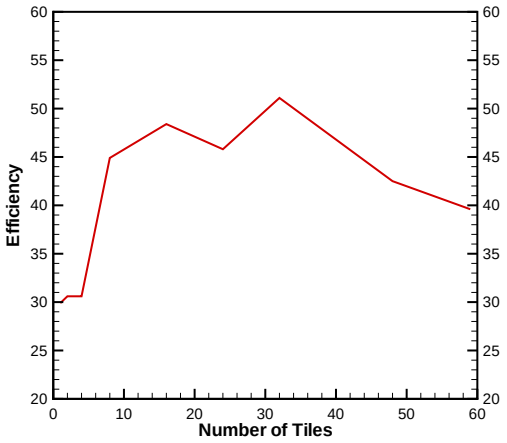
7. REFERENCES

[1] `http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/`.

[2] D. Abts, N. D. E. Jerger, J. Kim, D. Gibson, and M. H. Lipasti. Achieving predictable performance through better memory controller placement in many-core cmps. In *the* 36th *annual international*



(a)



(b)

Figure 8: Smith-Waterman benchmark (a) power (W) and (b) energy efficiency compared to a single core of an AMD quad-core Phenom II X4

symposium on Computer architecture, ISCA Š09, pages 451 – 461, New York, NY, USA, 2009.

[3] M. Berezecki, E. Frachtenberg, M. Paleczny, and K. Steele. Many-core key-value store. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8, July 2011.

[4] B. Bornstein, T. Estlin, B. Clement, and P. Springer. Using a multicore processor for rover autonomous science. In *Aerospace Conference, 2011 IEEE*, pages 1–9, March 2011.

[5] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[6] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[7] C. Chen, J. B. Manzano, G. Gan, G. R. Gao, and V. Sarkar. A study of a software cache implementation of the openmp memory model for multicore and manycore architectures. In *the* 16th *international Euro-Par conference on Parallel processing: Part II*, Euro-Par'10, pages 341 – 352, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] I. Choi, M. Zhao, X. Yang, and D. Yeung. Experience with improving distributed shared cache performance on tilera's tile processor. *IEEE Computer Architecture Letters*, 10(2):45–48, July 2011.

[9] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[10] J. Ha and S. P. Crago. Opportunities for concurrent dynamic analysis with explicit inter-core communication. In *the* 9th *ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, PASTE Š10*, pages 17 – 20, 2010.

[11] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[12] C. Hernández, A. Roca, J. Flich, F. Silla, and J. Duato. Characterizing the impact of process variation on 45 nm noc-based cmps. *J. Parallel Distrib. Comput*, 71:651–663, 2011.

[13] L. Karam, I. AlKamal, A. Gatherer, G. Frantz, D. Anderson, and B. Evans. Trends in multicore dsp platforms. *Signal Processing Magazine, IEEE*, 26(6):38–49, November 2009.

[14] A. Khajeh-Saeed and J. B. Perot. Gpu-supercomputer acceleration of pattern matching. In W.-M. W. Hwu, editor, *GPU Computing Gems*, chapter 13, pages 185–198. Morgan Kaufmann, emerald edition, 2011.

[15] A. Khajeh-Saeed, S. Poole, and J. B. Perot. Acceleration of the smith-waterman algorithm using single and multiple graphics processors. *Journal of Computational Physics*, 229:4247 – 4258, 2010.

[16] L. Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[17] J. Richardson, C. Massie, H. Lam, K. Gosrani, and A. George. Space applications on tilera. In *Workshop for Multicore Processors For Space - Opportunities and hallenges, IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, pages 19–23, Pasadena, CA, July 2009.

[18] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.

[19] TILERA Corporation. *Multicore Development Environment System ProgrammerŠs Guide*, June 2010.

[20] TILERA Corporation. *TILEPro64 Processor Data Sheet*, 2010.

[21] C. Ulmer, M. Gokhale, B. Gallagher, P. Top, and T. Eliassi-Rad. Massively parallel acceleration of a document-similarity classifier to detect web attacks. *J. Parallel Distrib. Comput*, 71:225–235, 2011.

[22] D. G. Waddington, C. Tian, and K. Sivaramakrishnan. Scalable lightweight task management for mimd processor. In *Systems for Future Multicore Architectures , EuroSys workshop*, pages 1–6, Salzburg, Austria, April 2011.

[23] C. Yan, F. Dai, and Y. Zhang. Parallel deblocking filter for h.264/avc on the tilera many-core systems. *In Advances in Multimedia Modeling*, 6523:51–61, 2011.