Numerical Algorithms, Libraries, and Software Frameworks for Future HPC Systems (Towards the Post Moore Era)

Takeshi Iwashita
Information Initiative Center, Hokkaido University, Sapporo, Japan
Email: iwashita@iic.hokudai.ac.jp

I. Introduction

Currently, the growth in the performance of supercomputers or high-end computing systems relies mainly on the increase in parallelism provided by many cores and special instruction sets. Consequently, researchers and developers of numerical algorithms and libraries must consider massive parallelism. At least $O(10^3)$ threads and $O(10^5)$ computational nodes should be effectively utilized. This will be a primary concern when developing novel algorithms and implementation methods for systems over the next few years. However, the situation will change within 10 years. It is predicted that Moore's law will end between 2025 and 2030.

When Moore's law ends, we will face a turning point. Although it is apparent that we cannot further improve the flops of a single chip, it is hard to forecast future computer and processor architectures. However, bytes will continue to increase. For example, three dimensional stacking and silicon photonics technologies will contribute to increases in the bandwidth between memory and processors, and between computational nodes. Moreover, non-volatile memory will be used to reduce power requirements.

However, to take full advantage of the benefits of the increase in bytes provided by these technologies in practical analyses, we must change our computational paradigm and numerical algorithms. Over the last decade, algorithms that use more flops and less bytes have been preferable, but now we must focus on increasing bytes but decreasing flops. However, efficient use of these new technologies is not straightforward. For future algorithms, we should consider complex and deep memory hierarchies, the heterogeneity of memory latencies, and the efficient use of logical units attached to memory modules. For example, we should intensively investigate bandwidth and latency reducing algorithms, which make more use of lower layers with higher memory bandwidths or reduce global synchronizations and communications.

In real applications, flops per watt is more important than flops. Many real world applications including bid data analyses rely on improvements in the flops per watt to lead to social innovations. Fortunately, the flops per watt can be improved even if Moore's law ends and the number of transistors that can operate for a fixed power input does not increase. For specific applications or computational kernels, we can effectively use special instructions (e.g., SIMD), accelerators, and reconfigurable hardware (e.g., FPGA) to increase the (effective) flops per watt. We should investigate novel implementation methods for these hardware systems and associated algorithms for the typical computational kernels required by real world applications. This research could also help to reduce the power consumption in real applications running on current or near future systems.

II. Objectives

Considering the above discussion, we have developed algorithms, libraries, and frameworks that effectively utilize massive parallelism, increases in data transfer rates, and low power systems.

In our research, we focused on four computational kernels: (1) iterative stencil computations; (2) transient analyses; (3) approximate matrix computations; and (4) sparse matrix computations. These kernels are used in various simulations such as plasma simulations, computational fluid dynamics, earthquake simulations, social system analyses, information sciences, data analyses, and electromagnetic field analyses.

A. Iterative stencil computations

In iterative stencil computations, we focused on the three dimensional finite difference time domain (FDTD) method, which is the most popular method for the electromagnetic field wave simulations. The 3D FDTD has a more complex stencil structure than the 7 point finite difference method. We have successfully applied temporal tiling to the method and also developed an auto-tuning method for the tile shape [1]. We are currently developing an implementation of the tiled 3D FDTD method that uses many core processors and GPUs, which must effectively use more than hundreds of threads.

B. High performance parallel multigrid solver in space and time for transient analyses

To accelerate transient analyses, we considered a parallel multigrid method in time. In [2], we developed a parallel two-level multigrid method in time for the non-linear finite element analyses of electric motors. Our method is more parallelized than the conventional parallel method in space. We are continuing this research by considering the "algebraic"-type parallel multigrid method in time.

C. Approximate matrix computations

We have paid special attention to H-matrices. An H-matrix is an approximation of a dense matrix. H-matrix methods reduce computational costs and memory requirements when applied to dense matrices. Many researchers have successfully applied this technique to various applications such as boundary element analyses or N-body problems. The H-matrix has a similar function to the fast multipole method (FMM). However, the FMM is a low B/F method and the H-matrix is a relatively high B/F method, so it is worth investigating for the system in the post Moore's law era. We have already developed a distributed parallel H-matrix library called "HACApK" with a software framework for BEM analyses [3] [4]. The HACApK library supports hybrid multi-processes and threads parallelism, but we will further improve it so that applications can run a huge number of processes and threads. Moreover, the library will be tuned for accelerators and future machines.

D. Sparse matrix computations and linear solvers

Sparse matrix computations are used in conventional PDE analyses and various new types of simulations such as those used to analyze big data. We investigated sparse matrix vector multiplication kernels, linear iterative solvers, and eigenvalue solvers. We will attempt to develop new algorithms and implementation methods for these kernels on many-core processors, accelerators, FPGAs, and computing units with associated memory.

In [5], we proposed a new fill-in strategy for incomplete Cholesky factorization preconditioning. Using this technique, nonzero blocks are defined for a coefficient matrix, and fill-ins are allowed in the blocks. Consequently, the preconditioning steps (forward and backward substitutions) consist of small dense matrix computations that are efficiently performed using SIMD instructions. Fig. 1 shows the vectorized implementation of multiplication of a small 2 by 2 matrix and a vector using intrinsic functions for SIMD instructions. Table I lists the numerical results using test matrices from the UF matrix collection, where the algebraic block multi-color ordering is used for parallelization of IC preconditioning. Our proposed preconditioning technique, ICB, mostly performs better than the conventional IC(0) preconditioning method because of its improved convergence and SIMD vectorization.

```
// Compute Ap=q
__m128d XA0, XA1, XP0, XP1, XQ0;
Iofs=0;
Vofs=0;
for(k=0; k<Nd; k++){
  XQ0  = _mm_setzero_pd();
  for(j=brow_ptr[k]; j<brow_ptr[k+1]; j++){
    Jofs = bcol_ind[j]*2;
    XP0 = _mm_load1_pd(&p[Jofs]);
    XP1 = _mm_load1_pd(&p[Jofs+1]);
    XA0 = _mm_load_pd(&bval[Vofs]);
    XA1 = _mm_load_pd(&bval[Vofs+2]);
    XA0 = _mm_mul_pd(XA0,XP0);
    XQ0 = _mm_add_pd(XQ0,XA0);
    XA1 = _mm_mul_pd(XA1,XP1);
    XQ0 = _mm_add_pd(XQ0,XA1);
    Vofs+=4;
  }
 _mm_store_pd(&q[Iofs],XQ0);
 Iofs+=2;
}
```

Figure 1. SIMD vectorized small 2 by 2 dense matrix vector multiplication using intrinsic functions

Table I
NUMERICAL RESULTS OF THE PARALLELIZED IC(0) AND ICB PRECONDITIONED CG SOLVERS BASED ON THE ABMC METHOD

Dataset name	Preconditioning	Number of # iteration	Elapsed time (s)	Elapsed time per iteration (s)
G3_circuit	IC(0)	256	2.38	9.28E-3
	ICB(2×2)	**206**	**2.11**	1.02E-2
Flan_1565	IC(0)	1778	93.83	5.28E-2
	ICB(4×1)	**1631**	**72.31**	4.43E-2
Hook_1498	IC(0)	1405	43.01	3.06E-2
	ICB(2×2)	**1347**	**35.33**	2.62E-2
thermal2	IC(0)	1616	**12.00**	8.65E-3
	ICB(2×1)	**1517**	12.20	8.04E-3
parabolic_fem	IC(0)	662	2.49	3.76E-3
	ICB(4×1)	**566**	**2.28**	4.03E-3

REFERENCES

[1] Takeshi Minami, Motoharu Hibino, Tasuku Hiraishi, Takeshi Iwashita and Hiroshi Nakashima, "Automatic Parameter Tuning of Three-Dimensional Tiled FDTD Kernel", in Proceedings of the Ninth International Workshop on Automatic Performance Tuning (iWAPT2014), (July 2014), Eugene, USA.

[2] Yasuhito Takahashi, Tadashi Tokumasu, Koji Fujiwara, Takeshi Iwashita, and Hiroshi Nakashima, "Parallel TP-EEC Method Based on Phase Conversion for Time-Periodic Nonlinear Magnetic Field Problems", IEEE Transaction on Magnetics, Vol. 51, No. 3, (2015), 7001305 (5 pages).

[3] Akihiro Ida, Takeshi Iwashita, Takeshi Mifune, and Yasuhito Takahashi, "Parallel Hierarchical Matrices with Adaptive Cross Approxima-tion on Symmetric Multiprocessing Clusters", Journal of Information Processing, vol. 22, (2014), pp. 642–650.

[4] Akihiro Ida, Takeshi Iwashita, Makiko Ohtani and Kazuro Hirahara, "Improvement of Hierarchical Matrices with Adaptive Cross Approximation for Large-scale Simulation", Journal of Information Processing, vol. 23, No. 3, (2015), pp. 366–372.

[5] Takeshi Iwashita, Naokazu Takemura, Akihiro Ida and Hiroshi Nakashima, "A new fill-in strategy for IC factorization preconditioning considering SIMD instructions", in Proceedings of the 13th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA-15), (August 2015), Helsinki, Finland, pp. 37-44.