Performance Characteristics of a Cosmology Package on Leading HPC Architectures

Jonathan Carter, Julian Borrill, and Leonid Oliker

CRD/NERSC
Lawrence Berkeley National Laboratory,
Berkeley, CA 94720
{jtcarter,jdborrill,loliker}@lbl.gov

Abstract. The Cosmic Microwave Background (CMB) is a snapshot of the Universe some 400,000 years after the Big Bang. The pattern of anisotropies in the CMB carries a wealth of information about the fundamental parameters of cosmology. Extracting this information is an extremely computationally expensive endeavor, requiring massively parallel computers and software packages capable of exploiting them. One such package is the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP) which has been used to analyze data from a number of CMB experiments. In this work, we compare MADCAP performance on the vector-based Earth Simulator (ES) and Cray X1 architectures and two leading superscalar systems, the IBM Power3 and Power4. Our results highlight the complex interplay between the problem size, architectural paradigm, interconnect, and vendor-supplied numerical libraries, while isolating the I/O filesystem as the key bottleneck across all the platforms.

1 Introduction

About 400,000 years after the Big Bang the expansion of space had cooled the Universe sufficiently for the charged electrons and protons to combine into neutral hydrogen atoms. At this point the primordial photons, which had been scattering off the free electrons, were suddenly able to propagate undisturbed through space, carrying with them a record of this moment which we call the Cosmic Microwave Background. The details of this snapshot — tiny variations in the photons' temperatures and polarizations — are an exquisitely sensitive probe of the fundamental parameters of cosmology, and measuring the detailed statistical properties of the CMB has been a high priority ever since its serendipitous discovery in 1965. The challenge lies in the fact that the continued expansion of the Universe has reduced the mean temperature of the CMB from around 3000K at last-scattering to only 3K today, and the anisotropies whose statistics we want to determine are at the 10^{-5} level in temperature, and anticipated to be at the 10^{-6}–10^{-8} level in polarization.

Realizing the extraordinary scientific potential of the CMB requires making precise measurements of the microwave sky temperature over a significant fraction of the sky at very high resolution. Such measurements are made by scanning the sky for as long as possible with a cryogenically cooled telescope and as many microwave detectors as possible. The reduction of the resulting datasets—first to a pixelized sky map, and then

to an angular power spectrum—is a serious computational challenge, and one which is only getting worse with increasing dataset sizes, as we try to make ever more precise measurements. It is therefore critical to choose the optimal algorithmic approach and supercomputing platform; one approach is the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP) [1], which has been widely used on a variety of supercomputers.

Until recently, CMB analyses were performed almost exclusively on superscalar cache-based microprocessors, due to their generality, scalability, and cost-effectiveness. However, for many classes of applications, these architectural platforms suffer from a growing gap between their sustained performance and claimed peak capabilities. Recently, two innovative parallel-vector architectures have become available to the supercomputing community: the Japanese Earth Simulator (ES) and the Cray X1. In order to quantify what these modern vector capabilities offer to scientists that rely on numerical simulation and data analysis, it is critical to evaluate this architectural approach in the context of demanding scientific computing algorithms [2–6]. Our research team was the first international group to conduct a performance evaluation study of the Earth Simulator, currently the world's most powerful supercomputer [7]. As remote ES access is not available, the study was performed during the authors' visit to the Earth Simulator Center located in Kanazawa-ku, Yokohama, Japan in December 2003.

In this work, we compare MADCAP performance on the vector-based ES and X1 architectures and two leading superscalar systems, the IBM Power3 and Power4. Two of the architectures studied, the X1 and Power4, were only available as relatively small systems. This restricted the size of problem that we were able to use for the comparison to the correspondingly small (15,000 pixel) CMB dataset from the MAXIMA balloon-borne experiment [8] on at most 64 processors. However, MADCAP's algorithmic development has been targeted at analyzing much larger datasets on many more processors. In particular, the recent introduction of gang-parallelism has enabled the dominant component of MADCAP—a set of independent dense matrix-matrix multiplications—to achieve near perfect scaling for a 100,000 pixel dataset on 1024, 2048, 3072 and 4096 Power3 processors. The results here show that the overheads associated with implementing this optimization negate most of the performance benefits for our experimental data set. Our analysis highlights the complex interplay between the problem size, architectural paradigms, interconnect fabric, and vendor-supplied numerical libraries, while isolating the I/O filesystem as the key bottleneck across the suite of HPC platforms.

2 Architectural Platforms

Table 1 presents a summary of the architectural characteristics of the four supercomputers examined in our study. Observe that the vector systems are designed with higher absolute performance and better architectural balance than the superscalar platforms. The ES and X1 have high memory bandwidth relative to peak CPU (bytes/flop), allowing them to continuously feed the arithmetic units with operands more effectively than the superscalar systems in our study. Additionally, the custom vector interconnects show superior characteristics in terms of measured latency [9, 10], point-to-point messaging (bandwidth per CPU), and all-to-all communication (bisection bandwidth) — in both raw performance (GB/s) and as a ratio of peak processing speed (bytes/flop).

Table 1. Architectural highlights of the Power3, Power4, ES, and X1 platforms

Platform	CPU/ Node	Clock (MHz)	Peak (GF/s)	Mem BW (GB/s)	Peak bytes/flop	MPI Lat (μsec)	Netwk BW (GB/s/CPU)	Bisect BW bytes/s/flop	Network Topology
Power3	16	375	1.5	0.7	0.47	16.3	0.13	0.087	Fat-tree
Power4	32	1300	5.2	2.3	0.44	12.0	0.06	0.012	Fat-tree
ES	8	500	8.0	32.0	4.0	5.6	1.5	0.19	Crossbar
X1	4	800	12.8	34.1	2.7	7.3	6.3	0.088[1]	2D-torus

The Power3 experiments reported here were conducted on the 380-node IBM pSeries system running AIX 5.1 and located at Lawrence Berkeley National Laboratory. Each 375 MHz processor contains two floating-point units (FPUs) that can issue a multiply-add (MADD) per cycle for a peak performance of 1.5 Gflop/s. Each SMP node consists of 16 processors connected to main memory via a crossbar. Multi-node configurations are networked via the SP Switch2 (Colony) switch using an omega-type topology. The IBM distributed filesystem, GPFS, was used for all benchmarks. The filesystem was configured with 16 GPFS servers (each 16 processor SMP nodes), each with 32GB of main memory that can be used to cache files and metadata. The total size of the filesystem was 30TB, with a block size of 256KB. In this model disk I/O uses the switch fabric, sharing bandwidth with message-passing traffic.

The Power4 experiments were performed on the 27-node IBM pSeries 690 system running AIX 5.2 and operated by Oak Ridge National Laboratory (ORNL). Each 32-way SMP consists of 16 Power4 chips (organized as 4 MCMs), where a chip contains two 1.3 GHz processor cores. Each core has two FPUs capable of a fused MADD per cycle, for a peak performance of 5.2 Gflop/s. Our benchmarks were run on a system employing the Colony interconnect. As in the Power3 case, GPFS was used for all benchmarks. The filesystem was configured with 8 GPFS servers (each a 4 CPU 1.7GHz Power4+) with 32GB of main memory. These servers support two 2TB filesystems, both with a block size of 256KB. The benchmarks utilized only one of these filesystems.

The 640 node ES runs enhanced Super-UX, a 64-bit Unix-based operating system. Each SMP node contains eight processors with 16 GB of memory, and are connected through a custom single-stage crossbar. The 500 MHz ES processor contains an 8-way replicated vector pipe (vector length = 256) capable of issuing a MADD each cycle, for a peak performance of 8.0 Gflop/s per CPU. For scalar instructions, the ES contains a 500 MHz scalar processor. Like traditional vector systems, the ES vector unit is a cache-less architecture; memory latencies are masked by overlapping pipelined vector operations with memory fetches. Each group of 16 nodes has a pool of RAID disk (720GB per node) attached via fiber channel switch. The filesystem used for our experiments is NEC's Supercomputer Filesystem (SFS), with a block size of 4MB. Each node has a separate filesystem, in contrast to the other architectures studied.

All Cray X1 benchmarks were performed on a 256-MSP system (several reserved for OS services) running UNICOS/mp 2.4 and operated by ORNL. The computational core, called the single-streaming processor (SSP), contains two vector (vector length = 64) pipes running at 800 MHz, giving a 3.2 Gflop/s peak for 64-bit data. The SSP also

[1] X1 bisection bandwidth is based on a 2048 MSP configuration

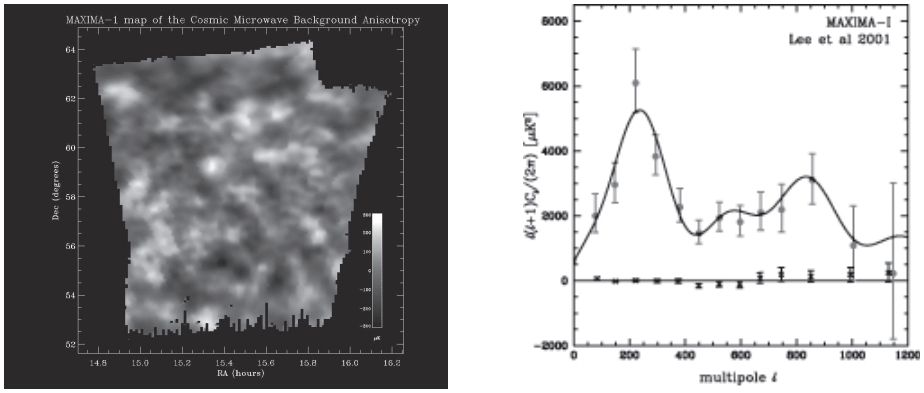

Fig. 1. The map and associated angular power spectrum of the part of the CMB sky measured by the MAXIMA experiment, as calculated by MADCAP.

contains a superscalar processor running at 400 MHz. The multi-streaming processor (MSP) combines four SSPs into one logical computational unit, sharing a 2MB data Ecache, that allows extremely high bandwidth (25–51 GB/s) for computations with temporal data locality. An X1 node consists of four MSPs sharing a flat memory, and large system configurations are networked through a modified 2D torus interconnect. The X1 at ORNL has four nodes available for I/O processing; each node is connected to a RAID array using fiber channel arbitrated loop protocol. Data transfer from a batch MSP must travel over the interconnect to one of the I/O nodes. The filesystem used in our study is a 4TB XFS filesystem, with a block size of 64KB.

3 MADCAP Overview

The analysis of a CMB dataset typically starts from the noise-dominated time-ordered data, constructs a pixelized map of the observed region (typically with signal-to-noise of around unity), and finally extracts the signal-dominated two-point angular correlation function, or power spectrum, of the CMB signal together with the errors on this spectral estimate (see Figure 1). The MADCAP approach is first to calculate the analytic maximum likelihood map and its residual pixel-pixel noise correlations, and then iteratively estimates the maximum likelihood power spectrum and its fisher information matrix. In this work we concentrate on the second step, which dominates the computational costs.

3.1 Methodology

The angular power spectrum is both a complete characterization of the CMB if its fluctuations are Gaussian, and is the statistic which can most readily be predicted for candidate cosmological models. MADCAP recasts the extraction of a CMB power spectrum from a map of the sky into a problem in dense linear algebra, and exploits the ScaLAPACK [11] libraries for its efficient parallel solution. The goal is to maximize the Gaussian log-likelihood of the data d (a pixelized sky map of dimension $\mathcal{N}_p$) over all possible

power spectrum multipole coefficients C_l where $\mathcal{L}(d|C_l) = -\frac{1}{2}\left(d^T\, D^{-1}\, d - \mathrm{Tr}\, \ln D\right)$ and D is the data correlation matrix $\langle dd^T \rangle$. In an ideal experiment, there would be an independent coefficient C_l for each multipole in the angular power spectrum. However, because of finite beam size and incomplete sky coverage, the accessible multipoles are instead grouped into $\mathcal{N}_b$ bins and a single coefficient C_b is associated with each bin.

Using Newton-Raphson iteration to locate the peak of this log-likelihood requires the evaluation of its first two derivatives with respect to the binned power spectrum coefficients. First, the data correlation matrix D is constructed as the sum of the experiment-specific noise correlations N and the theory-specific signal correlations $S(C_b)$ – then a square linear system $W_b = D^{-1}\frac{\partial S}{\partial C_b}$ is solved for each of the $\mathcal{N}_b$ spectral coefficients. This is accomplished by inversion of D and direct matrix-matrix multiplication. These operations scale as $\mathcal{N}_b\,\mathcal{N}_p^3$ for a map with $\mathcal{N}_p$ pixels. This number of pixels has progressively increased from $O(10^3)$ for the initial detection of CMB anisotropies by the COBE satellite to $O(10^4)$ – $O(10^5)$ for the ensuing ground- and balloon-based experiments, to $O(10^6)$ – $O(10^7)$ for the current WMAP and forthcoming Planck satellite missions.

MADCAP achieves its highest performance when the data are dense on the processors so that the communication overhead is minimized. With the advent of supercomputers with thousands of processors this was becoming harder to achieve for all but the largest datasets. MADCAP has therefore recently been rewritten to exploit the parallelism inherent in performing $\mathcal{N}_b$ independent matrix-matrix multiplications. The analysis is split into two steps: first, all of the processors collectively build and invert D; then, the processors are divided into independent gangs, each of which performs a subset of the multiplications. Since the matrices involved are block-cyclically distributed over the processors, this incurs the additional overhead of redistributing the matrices between the two steps. Our results compare these single- and multi-gang approaches.

3.2 Major Components

Each iteration of MADCAP's power-spectrum extraction algorithm is divided into seven steps. Table 2 presents an overview of the resource requirements. To maximize its ability to handle large datasets and many bins, MADCAP works with at most $3\mathcal{N}_p^2$ double words of memory, which corresponds to supporting 3 matrices in memory concurrently. The out-of-core disk-based storage for the other matrices in the calculation is the only practical choice given the number of bins, but comes at the cost of heavy I/O. All matrices are block-cyclic distributed across the processors; when a matrix is stored to disk due to memory limitations, MADCAP generates one file per processor over which the matrix is distributed. This results in matrix I/O operations that are independent, however the simultaneity of multi-processor disk accesses can create contention within the I/O infrastructure, thus degrading overall performance.

(i) **dSdC** calculates each of the pixel-pixel signal correlation derivative matrices dS/dC_b. The elements of these matrices are weighted sums of the Legendre functions P_l for each multipole l in bin b, evaluated for each pixel-pixel pair. Here $\mathcal{N}_b$ distributed matrices are output to disk. This step has high computational intensity on the superscalar architectures (over 7 flops/byte on the Power3/4), and medium on the vector machines, see Section 3.3 for details.

Table 2. Computational requirements for each iteration of MADCAP's power spectrum algorithm, in terms of pixels (N_p), bins (N_b), and multipoles (N_l)

Phase	Disk	RAM	Flops
dSdC	$8N_bN_p^2$	$16N_p^2$	$O(N_lN_p^2)$
invD	$8N_p^2$	$16N_p^2$	$O(N_p^3)$
redist	—	$16N_p^2$	—
W	$8N_p^2$	$24N_p^2$	$O(N_bN_p^3)$
dLdC	$8N_b$	$8N_p^2$	$O(N_bN_p^2)$
fisher	$8N_b^2$	$16N_p^2$	$O(N_b^2N_p^2)$
dC	$8N_b$	$8N_b^2$	$O(N_b^3)$
Total	$8(N_b+2)N_p^2$	$24N_p^2$	$O((N_b+1)N_p^3)$

(ii) **invD** calculates the full (symmetric positive definite) pixel-pixel data correlation matrix as $D = N + \sum_b C_b dS/dC_b$, explicitly inverts it using the ScaLAPACK `pdpotrf` and `pdpotri` routines, and performs the matrix-vector multiplication $z = D^{-1}d$ (where d is the data vector, the pixelized sky map) using the ScaLAPACK `pdsymv` routine. This step has an intermediate computational intensity of around 1.7 flops/byte. For our benchmarks we perform only the initial iteration in which $C_b = 0$, so that the $dSdC_b$ matrices do not actually need to be read in; this routine then reads and writes one matrix.

(iii) **redist**, one by one, reads each of the dS/dC_b matrices and the D^{-1} matrix, which are block-cyclic distributed across all of the processors, block-cyclic redistributes them using the `psgemr2d` routine, and rewrites them over one gang's worth of processors. This step performs no calculations per se, but is a set of data gathers by the group of processors in one gang from all other processors, stressing both memory bandwidth and system interconnect.

So far, all the processors have worked on the same step of the code (*single-gang* mode), since these operations are inherently sequential. The next steps may be performed in *multi-gang* mode. Note that *redist* step re-maps the data from single- to multi-gang mode and is not performed for single-gang MADCAP calculations.

(iv) **W** performs the multiplication $W_b = D^{-1}dS/dC_b$ for a given bin using the `pdgemm` routine. (Although D^{-1} and dS/dC_b are both positive definite symmetric matrices, dS/dC_b may have significant block structure which we can exploit by using `pdgemm` to multiply just the appropriate non-zero blocks, rather than using `pdsymm` on the whole matrices.) Depending on the amount of data per processor, intermediate to high computational intensity is required. This step requires each gang to read D^{-1} and its subset of the dS/dC_b matrices, and write the resulting W_b matrices.

(v) **dLdC** calculates the b^{th} element of the log-likelihood derivative vector using `pdgemv`, $dL/dC_b = z^T W_b d - Trace(W_b)$, since the W_b matrices are not symmetric. Each gang performs this multiplication for its subset of the bins. Matrix-vector multiply is of relatively low computational intensity, requiring an architectural balance between memory subsystem and peak arithmetic speed to achieve high performance. *dLdC* requires matrix input (W_b) but has no matrix output requirements.

(vi) **fisher** computes the b^{th} column of the bin-bin fisher matrix by first reading and transposing W_b, followed by reading W_b' for all $b' > b$ and calculating the trace as the

sum over all matrix element pair products: $F_{bb'} = Trace(W_b W_b')$. For the case where the number of bins exceeds the number of gangs, this step is load-balanced by giving each gang both a low and high numbered bin. In the case where number of gangs equal to the number of bins, there is an inherent load imbalance. The gang that processes the first bin will take on the order of $\mathcal{N}_b$ longer than the gang that processes the last bin. This step has low computational intensity, the main computational work being BLAS1; it has heavy I/O requirements, reading $\mathcal{N}_b(\mathcal{N}_b + 1)/2$ matrices.

(vii) **dC** calculates the correction $dC_b = F^{-1}_{bb'} dL/dC'_b$ using Cholesky decomposition of F and triangular solve. The number of bins is small enough that this is a simple serial code using the `dposv` and `dpotri` routines. We do not present an analysis of the *dC* phase, as it requires a trivial amount of runtime.

3.3 Vectorization

Most of the MADCAP routines utilize the ScaLAPACK library, making code migration a relatively simple task. Performance for both scalar and vector systems depends heavily on an efficient implementation of the vendor-supplied linear-algebra libraries. However, explicit vectorization was required for the hand-coded *dSdC* routine. The basic structure of the *dSdC* routine loops over all pixel pairs, calculating the value of Legendre polynomials up to some preset degree for the angular separation between these pixels. On superscalar architectures this constituted a largely insignificant amount of work, but due to the recursive computation, vectorization was prevented—resulting in significant overheads on the ES and X1. This routine was therefore rewritten so that at each iteration of the recursion a large batch of angular separations was computed in an inner loop. Compiler directives were required to ensure vectorization for both vector architectures. For our test case a speedup of approximately 10X and 30X were recorded on the ES and X1 respectively, bringing back a performance balance similar to the superscalar architectures.

3.4 Experimental Data

The data used in our experiments was collected by MAXIMA [8] (Millimeter Anisotropy eXperiment Imaging Array): a balloon-borne millimeter-wave telescope designed to measure the angular power spectrum of fluctuations in the CMB over a wide range of angular scales. MAXIMA has an unprecedented combination of sensitivity, angular resolution, and control of systematic effects. The experiment consists of a 1.3 m diameter off-axis Gregorian telescope and a receiver with a 16 element array of bolometers cooled to 100 mK. The high sensitivity of this receiver allows accurate measurements of the CMB power spectrum in a single overnight balloon flight. Each of the detectors in the array is sensitive to a single frequency band centered at 150, 240, or 410 GHz. The 150 GHz band is the most sensitive to the CMB and is close in frequency to the predicted minimum in galactic foregrounds. The higher frequency channels monitor emission from the atmosphere and galactic foregrounds such as dust.

4 Results

Our experiment used a dataset of 14996 pixels (N_p), 16 bins (N_b), and 1200 multipoles (N_l). We explore both single-gang (SG) runs, where all processors participate in each step of the calculation, and multi-gang (MG) runs, where gangs of processors carry out the $\mathcal{N}_b$ *W*, *dLdC*, and *fisher* steps concurrently. For each architecture we perform SG calculations using both 16 and 64 processors (SG processor counts are restricted to squared integers). The MG implementation depends on fast file-level synchronization across tasks. As the ES architecture provides this via MPI-IO or vendor-specific API (currently not utilized in MADCAP) and our short stay at the ES Center prevented code re-engineering, no MG experiments were performed. For all other architectures we performed MG calculations using 16, 32, and 64 processors with 4, 8, and 16 gangs of 4 processors respectively.

Tables 3-8 present a performance breakdown of MADCAP's five key steps. To distinguish between computational overhead and I/O requirements, we present two sets of runtime data (RT) for each experiments: the overall time (in wall-clock seconds); and the computational costs, without accounting for I/O operations and barrier wait times caused by I/O imbalance. For the ES, we were unable to measure the I/O and barrier times, so only the overall time is shown; we plan to gather these measurements on our next visit to the ES center. In addition, the parallel efficiencies (PE) of scaling from 16 to 64 processors are shown (P=32 is also presented for the MG case). Finally, we show the percentage of time each step accounts for in the overall MADCAP simulation (OT).

Recall that for *dSdC*, the SG and MG configurations are equivalent (exactly the same code is executed in both cases). We expect good scaling for this step, since it is embarrassingly parallel: The pixel map is divided amongst the processors and correlation matrix is computed independently, pixel by pixel. As the per-processor data density decreases at higher concurrencies, we expect to see a slight performance degradation due to loop overhead and decreased vector lengths.

Table 3 presents *dSdC* results, showing that the ES achieves the fastest raw performance, approximately 5x, 3.5x, and 2x faster than the Power3, Power4, and X1 respectively. For the Power3 and ES systems we see excellent speedup, but for the Power4 and X1 this is not the case. On further investigation, we determined that the main cause of slowdown was increased I/O time for writing out the matrix. Both the Power4 and

Table 3. Performance of *dSdC* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

dSdC		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	743	746	375	371	179	190	156	716	714	339	339	171	174
32	RT	—	373	—	199	—	97	—	—	359	—	172	—	93
	PE	—	100%	—	93%	—	98%	—	—	100%	—	98%	—	94%
64	RT	188	187	130	131	72	72	37	180	180	86	86	49	49
	PE	99%	100%	72%	70%	63%	66%	105%	100%	99%	98%	98%	88%	88%
	OT	7%	7%	6%	6%	8%	9%	4%	7%	7%	10%	11%	13%	12%

Table 4. Performance of *invD* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

invD		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	395	389	141	141	71	78	74	378	373	130	132	69	72
32	RT	—	213	—	90	—	52	—	—	204	—	81	—	44
	PE	—	91%	—	78%	—	75%	—	—	91%	—	81%	—	82%
64	RT	131	131	77	70	81	80	22	126	125	67	62	71	75
	PE	75%	74%	46%	50%	22%	24%	84%	75%	74%	49%	53%	24%	24%
	OT	5%	4%	4%	3%	9%	10%	2%	5%	5%	6%	6%	14%	13%

X1 systems have global filesystems with compute nodes having no direct connections to the I/O subsystem. We would expect to see some contention when increasing the number of I/O streams above a certain level, and this effect is most likely the reason for the slowdown. On removing the I/O time from the comparison the efficiency increases markedly, in line with our expectations.

For *invD*, we expect some slowdown on increasing the processor count, due the the decreased ratio of computation to communication; performance is presented in Table 4. As in *dSdC*, we note that the ES has the best parallel efficiency and absolute performance. Similarly, the Power3 has the second best efficiency, and is also the architecture where I/O scaling has the least impact. The Power4 and X1 both scale poorly, and removing I/O does not improve performance. For the Power4, we are comparing results between 16 processors (a half-populated SMP node) and 64 processors (two full SMP nodes), where the former case involves no intra-node communication and has twice the memory bandwidth per processor than in the latter case. For the X1, the scaling does not seem related to I/O, nor to the vector length (which remains similar going from 16 to 64 processors) — this issue is currently under investigation.

The *redist* step, presented in Table 5 taxes both memory bandwidth and interconnect efficacy. In addition, any low-level support for strided gathers and the ability of the ScaLAPACK to effectively use these features will also affect performance. Observe that this operation is fastest on the X1, while slowest on the Power4. However, since

Table 5. Performance of *redist* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

redist		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	0	461	0	280	0	254	0	0	456	0	273	0	252
32	RT	—	366	—	409	—	185	—	—	360	—	400	—	184
	PE	—	63%	—	34%	—	69%	—	—	63%	—	34%	—	69%
64	RT	0	222	0	296	0	186	0	0	217	0	289	0	185
	PE	—	52%	—	24%	—	26%	—	—	53%	—	24%	—	34%
	OT	0%	6%	0%	13%	0%	23%	0%	0%	9%	0%	23%	0%	30%

Table 6. Performance of *W* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

W		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	8501	7847	3176	2818	958	906	1345	7928	7474	2865	2582	791	693
32	RT	—	4052	—	1811	—	537	—	—	3753	—	1531	—	413
	PE	—	97%	—	78%	—	84%	—	—	100%	—	84%	—	84%
64	RT	2204	2029	1173	930	421	280	357	2066	1873	972	727	323	207
	PE	96%	97%	68%	76%	57%	81%	94%	96%	100%	74%	89%	61%	84%
	OT	79%	56%	53%	41%	48%	34%	37%	79%	81%	89%	72%	74%	41%

the X1 is has significantly higher peak performance than the superscalar systems in our study, *redist* accounts for a significant fraction of the X1's overall time (23%), and shows little parallel efficiency (26%). The cost of this step is vital to the success or failure of the multi-gang strategy: If it is too high, we will not recoup the loss via faster matrix-matrix multiplies in step *W*.

Before discussing the following set of results, we note that close to perfect parallel efficiency is expected for the multi-gang experiments. In both the 16 and 64 processor experiments, the computations and matrix distributions are identical — the only difference being that in the 64-way run, four matrix-matrix multiplies are performed concurrently. Table 6 shows the performance of *W* on our suite of architectural platforms. Observe that, as expected, the MG strategy reduces the overhead of *W* in all test cases when compared with the SG approach. The Power3 performs the closest to ideal scalability, particularly when I/O times are removed. This is followed by the Power4 and the X1. Note, however, that significant variability was seen in the X1's I/O performance, sometime by up to a factor of 4x. Observe that the X1 has the faster MG raw performance, achieving a 4.8x speedup over the Power3; however, it is also important to recall that the X1 is 8.5x faster in peak compared with the Power3.

For the *dLdC* step, shown in Table 7, we again expect high efficiency in MG mode. However, unlike step *W*, *dLdC* is dominated by I/O processing. The resulting computation-only runtimes are too small to clearly see the benefits of multi-gang parallelism,

Table 7. Performance of *dLdC* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

dLdC		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	284	276	111	132	95	86	166	15	19	7	9	5	1
32	RT	—	141	—	81	—	51	—	—	10	—	9	—	2
	PE	—	98%	—	81%	—	85%	—	—	98%	—	48%	—	37%
64	RT	73	71	62	78	87	52	48	8	4	5	5	6	9
	PE	98%	98%	45%	42%	27%	42%	86%	50%	121%	34%	46%	20%	4%
	OT	3%	2%	3%	3%	10%	6%	5%	3%	3%	5%	6%	15%	8%

Table 8. Performance of *fisher* using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), and as percentage of MADCAP's overall runtime (OT)

fisher		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	1361	1719	936	853	291	47	1442	591	489	125	104	63	28
32	RT	—	863	—	699	—	21	—	—	223	—	61	—	15
	PE	—	100%	—	61%	—	109%	—	—	110%	—	86%	—	93%
64	RT	589	880	653	694	85	16	416	383	72	149	38	47	10
	PE	58%	49%	36%	31%	86%	73%	87%	39%	170%	21%	69%	33%	71%
	OT	21%	24%	30%	31%	10%	2%	44%	21%	16%	49%	38%	15%	2%

except perhaps for the Power3. The total overhead times are influenced strongly by the ability of the filesystem to handle concurrent streams of I/O without loss of efficiency.

Results for the *fisher* step are shown in Table 8. As previously mentioned in Section 3.2, the 64-way MG experiment is inherently load imbalanced, due to the equal number of gangs and bins (16); thus, we expect poor scalability when comparing with the 16 processors simulation. For the Power3 and Power4, I/O accounts for a significant fraction of the runtime, while the X1 shows negligible I/O effects. In terms of runtime, the ES achieves surprising poor performance, almost five times slower than the X1 for the SG case — we plan to investigate this issue during our next visit to the ES Center.

5 Summary and Conclusions

Table 9 summarizes our findings by putting together all of MADCAP's components. We find that the X1 has the best runtimes: 1.1x, 2.8x, and 4.4x faster than the ES, Power4, and Power3 respectively; however, it suffers the lowest parallel efficiency. The ES and Power3 demonstrate the best scalability, significantly higher than the Power4 and X1. The Power3 shows the highest percentage of peak, followed by the ES, X1, and Power4; it is also the only architecture where the multi-gang strategy pays off for this dataset.

Our in-depth analysis of the performance of the MADCAP package demonstrates the complex interplay between the architectural paradigms, interconnect technology,

Table 9. Overall *MADCAP* performance using single-gang (SG) and multi-gang (MG): in runtime seconds (RT), parallel efficiency (PE), MFlop/s per CPU (MF/s/P), and percentage of peak .

MADCAP		Total overhead							Computation only					
P	metric	Power3		Power4		X1		ES	Power3		Power4		X1	
		SG	MG	SG	MG	SG	MG	SG	SG	MG	SG	MG	SG	MG
16	RT	11400	11522	4814	4646	1710	1672	3269	9688	9547	3506	3455	1171	1285
32	RT	—	6088	—	3344	—	1058	—	—	4924	—	2272	—	816
	PE	—	95%	—	69%	—	79%	—	—	97%	—	76%	—	79%
64	RT	3266	3606	2196	2264	873	823	954	2795	2492	1319	1240	567	624
	PE	87%	80%	55%	51%	49%	51%	86%	87%	96%	66%	70%	52%	52%
	MF/s/P	542	491	807	782	2029	2153	1857	634	711	1343	1429	3127	2840
	% peak	36%	33%	16%	15%	16%	17%	23%	42%	47%	26%	27%	24%	22%

and I/O filesystem. These design tradeoffs play a key role in algorithmic design and system acquisitions. Preliminary multi-gang parallel optimization has previously demonstrated high sustained performance for large problem sizes at extremely high concurrencies. However, for our experimental data set and limited processor count, little or no benefit was attained on a broad spectrum of supercomputers when using this optimized approach. Additionally, all evaluated architectural platforms sustained a relatively low overall fraction of peak, considering MADCAP's extensive use of computationally intensive dense linear-algebra calculations. Future work will examine higher-scalability simulations across a broad range of supercomputing systems, where we expect to cross the break-even point where multi-gang parallelism confers a clear performance advantage. We also plan investigate MADCAP's data transpositions and I/O transfer requirements in more detail, with the goal of reducing the impact of these overheads.

Acknowledgments

The authors would like to gratefully thank: the staff of the Earth Simulator Center, especially Dr. T. Sato, S. Kitawaki and Y. Tsuda, for their assistance during our visit; D. Parks and J. Snyder of NEC America for their help in porting applications to the ES; the MAXIMA team; and the NASA Advanced Information Systems Research Program which supported the development of MADCAP. IBM Power4 and Cray X1 access was graciously provided by ORNL; This research used resources of the NERSC at LBNL and CCS at ORNL supported by the DOE under Contract No DE-AC03-76SF00098 and DE-AC05-00OR22725 respectively. All authors from LBNL were supported by the Office of Advanced Scientific Computing Research in the DOE Office of Science under contract number DE-AC03-76SF00098.

References

1. Borrill, J.: MADCAP: The Microwave Anisotropy Dataset Computational Analysis Package. In: 5th European SGI/Cray MPP Workshop, Bologna, Italy (1999)
2. Agarwal, P.A., et al.: Cray X1 evaluation status report. In: Proc. of the 46th Cray User Group Conference, Knoxville, TN (2004)
3. Dunigan Jr., T.H., Fahey, M.R., III, J.B.W., Worley, P.H.: Early evaluation of the Cray X1. In: Proc. SC2003, Phoenix, AZ (2003)
4. Nakajima, K.: Three-level hybrid vs. flat mpi on the earth simulator: Parallel iterative solvers for finite-element method. In: Proc. 6th IMACS Symposium Iterative Methods in Scientific Computing. Volume 6. (2003)
5. Oliker, L., et al.: Evaluation of cache-based superscalar and cacheless vector architectures for scientific computations. In: Proc. SC2003, Phoenix, AZ (2003)
6. Oliker, L., et al.: A performance evaluation of the Cray X1 for scientific applications. In: 6th International Meeting on High Performance Computing for Computational Science. (2004)
7. Top500 Supercomputer Sites: (`http://www.top500.org`)
8. Hanany, S., et al.: Maxima-1: A measurement of the cosmic microwave background anisotropy on angular scales of $10'$–5°. The Astrophysical Journal **545** (2000) L5–L9
9. ORNL Cray X1 Evaluation: (`http://www.csm.ornl.gov/~dunigan/cray`)
10. Uehara, H., Tamura, M., Yokokawa, M.: MPI performance measurement on the Earth Simulator. Technical Report # 15, NEC Research and Development (2003/1)
11. The ScaLAPACK Project: (`http://www.netlib.org/scalapack/`)