Center for High-Performance Computing @ the University of Utah

News

University of Utah Volume 15, Number 1 Spring, 2004

Article

Programming Options for Distributed Shared Memory Cluster Computers

by Martin Cuma

Scientific Applications Programmer, Center for High Performance Computing, University of Utah

Parallel computers can be divided into two main categories: *shared memory processor* and *distributed-shared memory processor* machines.

Shared memory processor (SMP) machines enable multiple processors to address a single memory space. In distributed memory computers, memory allocated to each processor is local to the processor, which means no other processors can access that partition of memory. The data between the processors has to be communicated, most often via a message-passing library. Historically, SMP systems have been the most commonly used in parallel computing.

With the arrival of commodity clusters and portable message passing libraries, distributed memory computing has become increasingly popular. At present, we see a unifying trend: *distributed-shared memory processor* (DSMP) parallel computers that cluster together many multiprocessor nodes. The reasons for this trend include high hardware cost for large SMP machines, increased support for parallelism in processor design by major manufacturers (Intel, AMD) and lower per-processor cost of small SMP machines (2-4 CPUs) versus single processor (common mainboard, memory, hard drive) machines.

CHPC is increasingly moving towards this model of computing. About 40% of the Icebox cluster and the entire Arches cluster consist of dual processor SMP nodes. The Sierra cluster has four CPU SMP nodes.

Distributed and shared memory programming

Most of the parallel programs at present use *Message Passing Interface* (MPI)[1], which was designed to provide efficient and portable message passing for distributed memory computers. Although designed for distributed memory systems, MPI has been ported to most SMP computers as well.

The SMP performance of MPI, however, depends on its MPI implementation. MPI has to determine if the communication takes place within or outside of a node and select the most efficient communication device for each case. In order to communicate within the SMP node, MPI can either pretend that each of the two processes have distinct memory space and do a “communication” (which performs a memory copy from one location to another) or directly access the memory address at which the required variable resides. The latter approach is usually faster, although there is still a cost associated with the MPI library function call.

SMP machines commonly use a single process to run a parallel program and then parallelize the problem using threads. Threads execute within their parent process and share its memory space and stack (memory where subroutine data are allocated).

Threads can also have their specific, private data. Since the threads share many resources with their parent process, overhead associated with thread creation, management, and data access is lower than with processes.

Most common thread-based parallelization is performed using OpenMP[2] or Posix Pthreads[3]. While the latter is more complex and flexible, the majority of users prefer OpenMP for its simplicity and portability.

A logical, evolutionary approach on DSMP machines is *dual level parallelism*, which combines shared memory programming (such as with Pthreads or OpenMP) for intranode data transfer with MPI communication between the nodes. The aim of this article is to introduce the basics of mixed MPI/OpenMP programming and discuss when their use is appropriate.

Mixed MPI/OpenMP programming

Also called the *hybrid programming model*, message passing is used for communication across the nodes while OpenMP is used to access shared memory within the node. The goal is to provide better scalability than either MPI or OpenMP.

However, for each application, the user should evaluate if the performance gain of MPI/OpenMP code versus pure MPI outweighs the extra programming effort. In general, programs with little communication (sometimes referred to as *task level* or *embarrassingly parallel* programs) will not benefit greatly from the hybrid parallel model, while those with heavy inter-process communication (*domain level decomposed*) will most likely run faster. The best approach is to use MPI to parallelize on a large scale and to use OpenMP to break up computationally demanding loops within each MPI process.

The programming itself is relatively easy for those who know the basics of MPI and OpenMP: first, parallelize using MPI, then find loops that perform large amounts of computation and parallelize them using OpenMP. Since not all MPI implementations are thread safe, we recommend MPI communication only in the “serial” section of the program in which only one OpenMP thread is running per MPI process.

To demonstrate the bottom to top approach in MPI/OpenMP program development, we provide a simple program for the calculation of the π number using an easily parallelizable trapezoidal rule integral evaluation:

$$\int_0^1 \frac{dx}{x^2+1} = \frac{\pi}{4}$$

Figure 1 shows the serial implementation of the program. Not shown in the figure is a custom timer routine introduced in one of our previous newsletter articles[4]. We divide the integral into N sections, calculate the integral area sequentially, then print out the result and runtime.

```
#include <stdio.h>
#include <math.h>
#include “timer.h”

int main(int argc, char *argv[]){
const int N = 10000000000;
const double h = 1.0/N;
const double PI = 3.141592653589793238462643;
int i;
double x,sum,pi,error,time;

time = -ctimer();

sum = 0.0;

for (i=0;i<=N;i++){
  x = h * (double)i;
  sum += 4.0/(1.0+x*x);
}
pi = h*sum;

time += ctimer();

error = pi - PI;
error = error<0 ? -error:error;
printf(“pi = %18.16f +/- %18.16f\n”,pi,error);
printf(“time = %18.16f sec\n”,time);
return 0;
}
```

Figure 1 -- Serial version of pi code

Figure 2 shows the same program with OpenMP parallelization of the trapezoidal loop (in the code listing, “↲“ indicates a line continuation). Note that only one line needs to be added to the serial code. It includes the *parallel for* directive and specification to determine which of the data values are shared by threads and which are private. In addition, we have to reduce-sum the local value of the integral.

In order to run the OpenMP job in parallel, there are two requirements: first, it must be compiled with a compiler that supports OpenMP (which includes most of the commercial compilers) and the appropriate flag (usually *-mp* or *-omp*), then the number of threads to run must be specified. The easiest way to do this is to set the O*MP_NUM_THREADS* environment variable before program execution. For more details on how to run OpenMP programs on CHPC computers, consult our online course or help page[5,6].

```
#include <stdio.h>
#include <math.h>
#include “timer.h”

int main(int argc, char *argv[]){
const int N = 1000000000;
const double h = 1.0/N;
const double PI = 3.141592653589793238462643;
int i;
double x,sum,pi,error,time;

time = -ctimer();

sum = 0.0;

#pragma omp parallel for shared(N,h), ↲
       private(i,x),reduction(+:sum)

for (i=0;i<=N;i++){
  x = h * (double)i;
  sum += 4.0/(1.0+x*x);
}
pi = h*sum;

time += ctimer();

error = pi - PI;
error = error<0 ? -error:error;
printf(“pi = %18.16f +/- %18.16f\n”,pi,error);
printf(“time = %18.16f sec\n”,time);
return 0;
}
```

Figure 2 -- OpenMP version of pi code

The MPI version of the *pi* program is shown in Figure 3 (see page 3). The program is slightly more complicated due to the fact that we must explicitly divide the work among the parallel processes. For this, we call MPI functions that determine the number of processors and the processor index. We must also reduce-sum the local result.

The program is linked with the MPI libraries and run using the *mpirun* command. For details on how to do this on CHPC systems, see our online course or help page[7,8].

A final, hybrid MPI/OpenMP program is presented in Figure 4 (see page 3). It differs from the MPI version only in the insertion of the *parallel for* directive before the integral loop. This program has to be compiled both with *-mp* flag (using PGI compilers on Icebox) and with MPI libraries.

Before running the program, the user has to set the *OMP_NUM_THREADS* environment variable. This may cause problems on multiple node runs on Icebox because the environment is not passed to the nodes by default. A simple way to avoid these kinds of hassles is to set the number of threads explicitly in the program by calling the OpenMP *omp_set_num_threads()* function as shown in Figure 4. Because this is an external function call, we must include the *omp.h* header file in the program. Details on how to compile and run MPI/OpenMP programs are offered in our online course[9].

	Processors/threads total for the whole application						
	1/1	1/2	2/1in*	2/1out*	2/2	2/4	4/1
Serial (1)	14.037						
OpenMP (2)	13.942	6.986					
MPI (3)	14.035		7.021	7.025			3.513
MPI/OpenMP (4)	14.199	7.101			7.107	3.554	

* 2/1in denotes processors in a single node; 2/1out denotes 2 processors on 2 nodes.

Table 1. Timing of example program on Icebox 1533 MHz dual AthlonXP nodes in seconds.

Table 1 shows the timing of our example program. Several notes are appropriate:

1. First, the pure OpenMP program runs faster than pure MPI on the two processors of the SMP node. Creation and management of OpenMP threads consume fewer resources than in MPI processes.
2. As expected, a two processor MPI run inside of one node is slightly less time consuming than on two nodes. The in-node reduction operation is faster than the out-of-node reduction involving network communication between the two nodes.
3. Finally, and perhaps the most surprising, the mixed MPI/OpenMP program performs *worse* than either pure MPI or OpenMP code. The reason for this is probably the low granularity of our problem. The extra time caused by the scheduling overhead of the OpenMP parallel loop does not overcome the savings of not having to do single MPI reduction operations.

This demonstrates the fact that not every MPI application can benefit from threading on DSMPs. On the other hand, since MPI reduction operations tend to scale poorly with large numbers of processors, the MPI/OpenMP program performance may get closer to that of pure MPI on large numbers of processors.

```
#include <stdio.h>
#include <math.h>
#include “mpi.h”
#include “timer.h”

int main(int argc, char *argv[]){
const int N = 1000000000;
const double h = 1.0/N;
const double PI = 3.14159265358979323846264З;
int i;
double x,sum,pi,error,time,mypi;

int myrank,nproc;

MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
MPI_Comm_size(MPI_COMM_WORLD,&nproc);

time = -ctimer();

sum = 0.0;
for (i=myrank;i<=N;i=i+nproc){
  x = h * (double)i;
  sum += 4.0/(1.0+x*x);
}
mypi = h*sum;

MPI_Reduce(&mypi,&pi,1,MPI_DOUBLE,MPI_SUM,0, ↲
       MPI_COMM_WORLD);

time += ctimer();

error = pi - PI;
error = error<0 ? -error:error;
if (myrank==0){
printf(“pi = %18.16f +/- %18.16f\n”,pi,error);
}
printf(“proc %4d time = %18.16f sec\n”,myrank, ↲
       time);
MPI_Finalize();
return 0;
}
```

Figure 3 -- MPI version of pi code

```
#include <stdio.h>
#include <math.h>
#include <mpi.h>
#include <omp.h>
#include “timer.h”

int main(int argc, char *argv[]){
const int N = 1000000000;
const double h = 1.0/N;
const double PI = 3.141592653589793238462643;
int i;
double x,sum,pi,error,time,mypi;

int myrank,nproc;

MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
MPI_Comm_size(MPI_COMM_WORLD,&nproc);

time = -ctimer();

sum = 0.0;
omp_set_num_threads(2);

#pragma omp parallel for shared(N,h,myrank,nproc), ↲
       private(i,x),reduction(+:sum)
for (i=myrank;i<=N;i=i+nproc){
  x = h * (double)i;
  sum += 4.0/(1.0+x*x);
/*  printf(“%d %d %10.5f”,my_rank,i,x); */
}
mypi = h*sum;

MPI_Reduce(&mypi,&pi,1,MPI_DOUBLE,MPI_SUM,0, ↲
       MPI_COMM_WORLD);

time += ctimer();

error = pi - PI;
error = error<0 ? -error:error;
if (myrank==0){
printf(“pi = %18.16f +/- %18.16f\n”,pi,error);
}
printf(“proc %4d time = %18.16f sec\n”,myrank,time);
MPI_Finalize();
return 0;
}
```

Figure 4 -- MPI/OpenMP version of pi code

Conclusions

In this article, we introduced the concept of hybrid MPI/OpenMP programming on distributed shared memory computers. The goal was to reduce the program execution overhead on a multiprocessor node by changing from heavyweight processes to lightweight threads and by avoiding inter-process MPI communication function calls. As demonstrated in our examples, the introduction of OpenMP into existing MPI code is relatively straightforward. Distributed code that uses communication heavily will benefit the most; embarrassingly parallel programs that do a minimal amount of communication may even see decreases in performance due to OpenMP thread scheduling overhead.

References

[1] Message Passing Interface standard: http://www-unix.mcs.anl.gov/mpi/
[2] OpenMP standard: http://www.openmp.org/
[3] POSIX Pthreads API, ANSI/IEEE POSIX 1003.1c standard (1995), good tutorial: http://www.llnl.gov/computing/tutorials/workshops/workshop/pthreads/MAIN.html
[4] "Tips and tricks for programming on Icebox and Sierra part 2." Tools for timing user programs and smart ways to code for speed, CHPC Newsletter Winter 2003
[5] CHPC's C and Fortran help webpage: http://www.chpc.utah.edu/index.php?currentNumber=3.2.50, http://www.chpc.utah.edu/index.php?currentNumber=3.2.110
[6] CHPC's "Introduction to programming with OpenMP" course: http://www.chpc.utah.edu/general/short_courses/intro_omp/
[7] CHPC's MPI help webpage: http://www.chpc.utah.edu/index.php?currentNumber=3.2.200
[8] CHPC's "Introduction to programming with MPI" course: http://www.chpc.utah.edu/general/short_courses/intro_mpi/
[9] CHPC's "Hybrid MPI/OpenMP" course: http://www.chpc.utah.edu/general/short_courses/index.html

FYI

¤ CHPC presents "Introduction to CHPC" on the 1st Tuesday of every month. Please join us for our next one on April 6th at 1:00 p.m. in the INSCC Auditorium. Please see http://www.chpc.utah.edu/index.php?currentNumber=1.4 for more details.

¤ CHPC's seminars are back! For up to date information on schedules, descriptions, and for information on past seminars, check the CHPC seminar web page: http://www.chpc.utah.edu/~baites/seminars.html

¤ When creating works that require CHPC resources, please remember to cite CHPC's contributions in your final product. These citations play a vital role in our ability to continue providing these services!

Report

On the Scene: IEEE/ACM SuperComputing Conference 2003

by Sam Liston
Digital Communication & Visualization, Center for High Performance Computing, University of Utah

The University of Utah's participation in the IEEE/ACM SuperComputing Conference 2003 (a.k.a. "SC2003") was a great success. CHPC, in cooperation with the Scientific Computing and Imaging Institute (SCI) and the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), was able to put together a booth that effectively displayed our current work as well as showcase our future endeavors.

One such endeavor is the soon to be implemented NIH Bioinformatics Cluster named "Arches." CHPC had 48 nodes of the data mining portion of Arches on the exhibit floor, being used by researchers from the University of Utah and Los Alamos National Lab. Having such a large production machine on the floor of SC was a first for CHPC. It generated a good volume of traffic and interest.

AMD
ANGSTROM
is proud to announce
The University of Utah will soon have the largest AMD Opteron™ cluster with a whopping 500 servers!
Titan64 SuperBlade
World's Densest Server Rack

Some of this interest stemmed from a bit of extra and unexpected recognition at the AMD booth, where it was advertised that the University of Utah would soon have the largest total Opteron-based cluster. The additional traffic generated by this advertisment resulted in a good amount of Arches-related discussion and Q&A , making the cluster a valuable addition to the booth.

Many other CHPC-supported projects were represented in the booth as well. A number of video interviews and posters discussed topics such as meteorology, modern dance, telemediated arts, computational chemistry, and genomics. In the video interviews, individuals discussed their current research and theorized on the possible impacts of their work. These interviews were taped, captured and then compiled onto DVD.

Another integral part of the booth contained two large 8 ft. by 4 ft. screens, on which four projectors displayed images highlighting the University of Utah (campus, students, sports, etc) as well as visualizations from various groups showing off their research.

This was our most visible booth to date. Though its construction, which consisted of an exposed PVC pipe skeleton supported by paneled walls, had a rough "industrial" look to it, its towering 12 ft. structure and displays of constantly moving images were visible from a good portion of the conference hall.

This was the 15th anniversary of the annual conference and the first time for it was held in Phoenix, AZ. It was also the most successful to date. Overall attendance surpassed 7,600, exceeding last year's record by 300 attendees. This was also the largest concerning exhibits and space. The conference boasted 219 booths consisting of both industry and research exhibits, occupying nearly 100,000 square feet of floor space.

Some additional highlights of the conference came from an event known as the "Bandwidth Challenge". In this contest, teams are challenged to push as much data as possible in a set amount of time. The winners, a team from the Stanford Linear Accelerator and Los Alamos National Lab, were able to achieve a sustained network throughput of 23.21 Gb/sec., which beat out last years winners by 5 Gb/sec. The total amount of data that the team moved in the allotted time exceeded 6,500 Gb.

SuperComputing 2004 will be held in Pittsburgh, PA, at the new David L. Lawrence Convention Center. CHPC will be there, and hopes to have a bigger, better booth, representing and highlighting the work of an even larger number of its supported projects.

If you are interested in having your research highlighted in next year's booth you can contact Sam Liston at *stliston@chpc.utah.edu* for more information.

FYI

For the latest news, system status, and downtimes, see the CHPC home page: *http://www.chpc.utah.edu/*

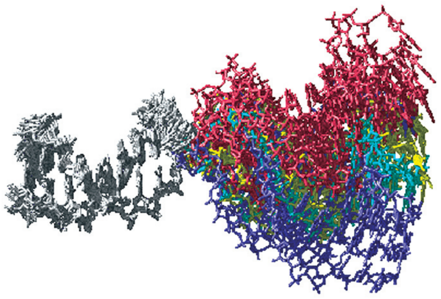
Report

On the Scene: Research Posters on the Hill 2004

by Robert McDermott
Staff Scientist, Visualization Group, Center for High Performance Computing, University of Utah

This year's Posters on the Hill event took place from 8:00am to 1:00pm on Thursday, January 22. A total of 30 posters were presented. All 19 women and 11 men who participated are graduates from high schools within the state of Utah.

Two focal points of this year's event were a print entitled "Sagebrush to Steel" by Stefanie Joos Dykes of the Art Department from the College of Fine Arts, and a formula racer designed, fabricated, and raced by a team of students under the direction of faculty member Sam Drake in the Mechanical Engineering Department and the School of Computing.


A visualization of how DNA bends and writhes (by Elijah Gregory, Thomas Cheatham, and Julio Facelli)

There were posters from across the University covering a wide variety of disciplines. From the College of Science, we had representation from Biology, Chemistry, Mathematics, and Physics. From the College of Engineering, we had Bioengineering, Chemical & Fuels Engineering, Electrical & Computer Engineering, Mechanical Engineering, and the School of Computing. The Medical Community provided posters from Human Genetics, Medicinal Chemistry, Ophthalmology, Orthopedics, Pediatrics, Pharmaceutics and Pharmaceutical Chemistry, and Oncological Sciences. The Center for High Performance Computing, the Cardiovascular Research and

Training Institute, the Graduate School of Architecture, the Art Department, and the Psychology Department had posters presented as well.

Interesting individual imagery was submitted by Tom Johnson and Steven Parker from the School of Computing; Elijah Gregory from the Deptartment of Medicinal Chemistry, Thomas Cheatham from the Department of Pharmaceutics and Pharmaceutical Chemistry, and Julio Facelli from the Center for High Performance Computing; and Amy Heaton and Ken Golden from the Department of Mathematics.

An example of how tone mapping retains color and brightness (by Tom Johnson and Steven Parker, School of Computing)

Visually striking posters were produced by Tracy Zundel and Katharine Ullman in Oncological Sciences; Shawn Olsen and Erik Jorgensen from Biology; and Harmony Hofstetter and Julio Bermudez from the School of Architecture.

Governor Olene Walker takes a moment to pose for some photographs with the formula racer

The highlight of this year's event was Governor Olene Walker, who took a break from her "State of the State" address preparations to view the posters and to pose for some photographs with the formula race car, which Sam Drake and seven of his students hand-carried up the front steps of the State Capitol building at 7:30am.

Full size, complete versions of the graphics shown in this report can be found in the "Research" section of the CHPC website: http://www.chpc.utah.edu/other/research/

Recent Publications

Archbold, Gregory C. "A Study of the Composition of Ultra High Energy Cosmic Rays Using the High Resolution Fly's Eye," Doctoral Dissertation, Dept. of Physics, University of Utah, August 2002.

Bazterra, Victor E., Marta B. Ferraro, and Julio Facelli. "Modified Genetic Algorithm to Model Crystal Structures: III. Determination of Crystal Structures Allowing Simultaneous Molecular Geometry Relaxation." International Journal of Quantum Chemistry, 96 312-320 (2004)

Blair, Steve. "Self-focusing of narrow 1-D beams in photonic microcavity arrays," Journal of the Optical Society of America B 20, p. 1520-1526 (2003)

Burns, Thomas J., Edward L. Kick and Byron L. Davis. "Theorizing Linkages Between the Natural Environment and the Modern World-System: Deforestation in the Late 20th Century." Journal of World Systems Research, IX:2 357-390 (Summer 2003).

Cheng, W.Y.Y. and W.J. Steenburgh, 2003: The Realtime Weather and Forecasting Research Model for the Western United States at the NOAA Cooperative Institute for Regional Prediction. Tenth Annual Workshop on Weather Prediction in the Intermountain West, Desert Research Institute, Reno, Nevada, November 6, 2003.

Facelli, Julio. "Calculations of Chemical Shielding: Theory and Applications." Concepts in Magnetic Resonance Part A, 20A:1 42-69 (2004)

Jenkins, M.A. "A examination of the sensitivity of numerically simulated wildfires to low-level atmospheric stability and moisture, and the consequences for the Haines Index." Int. J. of Wildland Fire 11:4, 213-23 (2002)

Jenkins, M. A. "Investigating the Haines Index using parallel model theory." Int. J. of Wildland Fire. (2004)

Jenkins, M. A., and Ruiyu Sun. "What coupled wildfire-atmosphere numerical models can do." 4th International Workshop on Disturbance Dynamics in Boreal Forest. Prince George, British Columbia, August, 2002.

Jenkins, M. A., S. K. Krueger, and R. Sun. "Using a simple parallel model to investigate the Haines Index." 5th Symposium on Fire and Forest Meteorology, American Meteorological Society, Orlando, FL. 2003.
(A summary of this work in the December 2003 Bulletin of the American Meteorological Society.)

Sun, Ruiyu. "Effect of low-level atmospheric stability and moisture on wildfire behavior." M. Sc. Thesis, University of Utah, 2003.

Sun, R., and M. A. Jenkins. "Effects of near-surface atmospheric stability and moisture on wildfire behavior and consequences for the Haines Index." 5th Symposium on Fire and Forest Meteorology, American Meteorological Society, Orlando, FL. 2003.

Sun, Ruiyu, M. A. Jenkins, and S. K. Krueger. "Effects of atmospheric stability, moisture, and wind speed on wildfire behavior and indications of Haines Index." Submitted for publication in special 5th Symposium on Fire and Forest Meteorology issue of the Int. J. of Wildland Fire. 2004.

Whitaker, J., E. Ahn, P. Hari, G. A. Williams, P. C. Taylor, and J. C. Facelli, "Indirect (J) coupling of inequivalent 75As nuclei in crystalline and glassy As2Se3 and As2Se3." Journal of Chemical Physics, 119:16 8519-8525 (2003)

CHPC Staff Directory

Administrative Staff	Title	Phone	Email	Location
Julio Facelli	Director	581-7529	Julio.Facelli@utah.edu	410 INSCC
Julia Harrison	Associate Director	581-5172	julia@chpc.utah.edu	430 INSCC
Guy Adams	Assistant Director, Systems	585-0471	gadams@chpc.utah.edu	424 INSCC
Joe Breen	Assistant Director, Networking	585-1013	jbreen@chpc.utah.edu	426 INSCC
DeeAnn Raynor	Administrative Officer	581-5253	dee@chpc.utah.edu	412 INSCC
Victoria Volcik	Administrative Assistant	585-3791	vicky@chpc.utah.edu	405-6 INSCC
David Weiland	Administrative Assistant	581-6440	weiland@chpc.utah.edu	405-2 INSCC

Scientific Staff	Expertise	Phone	Email	Location
James Agutter	Information Visualization	581-8779	agutterja@arch.utah.edu	235 AAC
Thomas Cheatham III	Biomolecular Modeling	587-9652	cheatham@chpc.utah.edu	306 INSCC
Martin Cuma	Scientific Applications	587-7770	mcuma@chpc.utah.edu	418 INSCC
Byron L. Davis	Statistics	585-5604	byron@chpc.utah.edu	416 INSCC
Julio Facelli	Molecular Sciences	581-7529	Julio.Facelli@utah.edu	410 INSCC
Stefano Foresti	Information Visualization	581-3173	stefano@chpc.utah.edu	312 INSCC
Robert McDermott	Visualization	581-4370	mcdermott@chpc.utah.edu	420 INSCC
Brett Milash	Biochemistry	585-6408	brett.milash@hci.utah.edu	SOM
Anita Orendt	Molecular Sciences	587-9434	orendt@chpc.utah.edu	422 INSCC
Alun Thomas	Bioinformatics	587-9309	alun@gene.pi.med.utah.edu	Research Park

Systems/Network Staff	Title	Phone	Email	Location
Irvin Allen	Desktop Support	581-7996	iallen@chpc.utah.edu	405-29 INSCC
Wayne Bradford	Cluster Grid Administrator	585-1333	wayne.bradford@chpc.utah.edu	405-40 INSCC
Erik Brown	Computation Cluster Admin.	581-3442	erik@chpc.utah.edu	405-31 INSCC
Joe Clyde	Network Operations Engineer	585-2548	joe.clyde@chpc.utah.edu	405-38 INSCC
Geoff Fritz	Enterprise Server Administrator	585-0567	gfritz@chpc.utah.edu	405-41 INSCC
Brian Haymore	Computation Cluster Admin.	585-1755	brian@chpc.utah.edu	428 INSCC
Samuel T. Liston	Digital Communication & Visualization	585-1577	stliston@chpc.utah.edu	405-30 INSCC
Jimmy Miklavcic	Multimedia, Telematic & Digital Communication	585-9335	jhm@chpc.utah.edu	296 INSCC
Ron Price	Cluster Grid Administrator	560-2305	rprice@eng.utah.edu	405-19 INSCC
David Richardson	Computer Technician	581-8646	drr@chpc.utah.edu	405-8 INSCC
Steve Smith	Desktop Support	581-7552	steve@chpc.utah.edu	405-14 INSCC
Matthew Thorley	Network Assistant	585-7821	ruach@chpc.utah.edu	405-20 INSCC
Kirk VanOpdorp	Computation Cluster Admin.	585-9299	kirk@chpc.utah.edu	405-31 INSCC
Alan Wisniewski	Network Assistant	580-5835	quantix@chpc.utah.edu	405-21 INSCC

User Services Staff	Title	Phone	Email	Location
Nathan Barker	Technical Specialist	N/A	barkern@chpc.utah.edu	294 INSCC
Eric Hansen	Technical Assistant	N/A	ehansen@chpc.utah.edu	405-27 INSCC
Shawn Lyons	Network Assistant	581-4439	slyons@chpc.utah.edu	405-22 INSCC
Murat Manguoglu	Technical Specialist	N/A	murat@chpc.utah.edu	405-11 INSCC
Beth Miklavcic	Multimedia Design, Digital Video	585-1067	bam@chpc.utah.edu	405-13 INSCC
Erik Ratcliffe	Graphic & Web Design	N/A	erat@chpc.utah.edu	405-14 INSCC
Jason Rino	Systems Assistant	581-4439	jason@chpc.utah.edu	405-22 INSCC

The University of Utah seeks to provide equal access to its programs, services, and activities to people with disabilities. Reasonable prior notice is needed to arrange accommodations.

Welcome to CHPC News!

If you would like to be added to our mailing list, please fill out this form and return it to:

Vicky Volcik
UNIVERSITY OF UTAH
Center For High Performance Computing
155 S 1452 E ROOM 405
SALT LAKE CITY, UT 84112-0190
FAX: (801)585-5366

(room 405 of the INSCC Building)

Name:
Phone:

Department or Affiliation:
Email:

Address:
(UofU campus or U.S. Mail)

Thank you for using our Systems!

Please help us to continue to provide you with access to cutting edge equipment.

ACKNOWLEDGEMENTS

If you use CHPC computer time or staff resources, we request that you acknowledge this in technical reports, publications, and dissertations. Here is an example of what we ask you to include in your acknowledgements:

"*A grant of computer time from the Center for High Performance Computing is gratefully acknowledged.*"

Please submit copies of dissertations, reports, preprints, and reprints in which the CHPC is acknowledged to: Center for High Performance Computing, 155 South 1452 East, Rm #405, University of Utah, Salt Lake City, Utah 84112-0190

UNIVERSITY OF UTAH
Center for High Performance Computing
155 South 1452 East, RM #405
SALT LAKE CITY, UT 84112-0190