Advanced Topics on Code Optimization

Masao Fujinaga
Research Computing Support
Academic Information and Communication Technology
University of Alberta

Connect to the cluster

- ssh -Y **ccid**@cluster.srv.ualberta.ca
- "-Y" : enables trusted X11 forwarding
- cd /scratch/**ccid**/
- cp -r /scratch/fujinaga/may2010-opt .
- cd may2010-opt
- ggv may2010-opt.pdf &

Steps in Optimization

- Check for correct results
- Profiling and analyzing the program
- Optimize I/O
- Increase memory use of older programs
- Compiler optimizations
- Optimizing cache use
- Replace or avoid slow operations
- Use tuned libraries
- Look for a better algorithm

Timing

- Reproducible
- Error
- Intrusive?
- Work from local filesystem
- Effect of other users

Timing (continued)

- time (shell command, /usr/bin/time)
  - User time
  - System time
  - Wall time
- System routines
  - dtime, etime (fortran)

```
real tarray(2), start, end, cpuTime
start = etime(tarray)
call doWork
end = etime(tarray)
cpuTime=end - start
```

Exercise 1

- cd /scratch/**ccid**
- cp -r /scratch/fujinaga/may2010-opt .
- cd may2010-opt
- pgf77 -o prob1 prob1.f (or pgcc -o prob1 prob1.c)
- qsub job.pbs
- qstat -u **ccid**
- cat out

Checksum: 362638531.249306

real	0m59.343s
user	0m57.332s
sys	0m0.644s

Profiling the program

- gprof

  pgf77 -o prob1 prob1.f -pg

  qsub job.pbs
  - Produces gmon.out

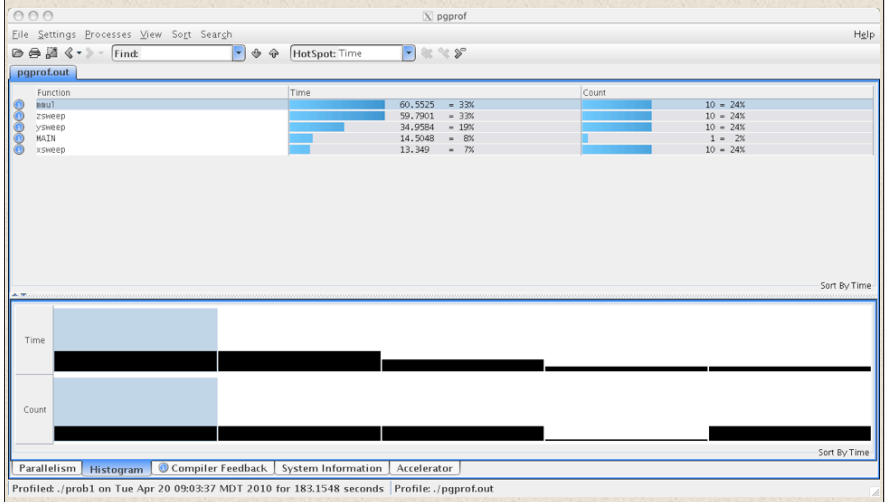
  gprof prob1
- pgprof

  pgf77 -o prob1 prob1.f -Mprof=line

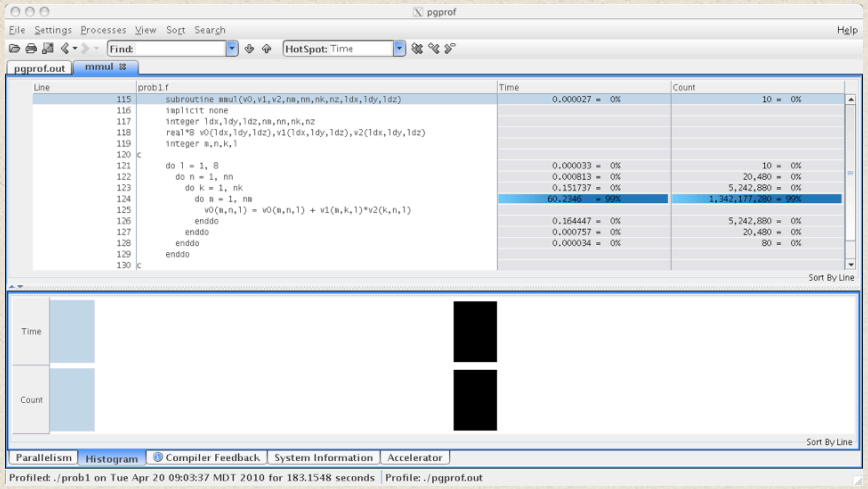
  qsub job.pbs
  - Produces pgprof.out

  pgprof -exe prob1 pgprof.out

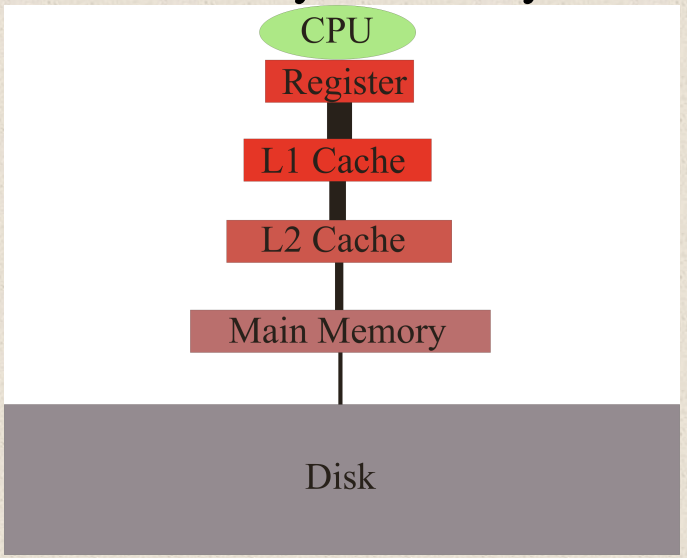
pgprof



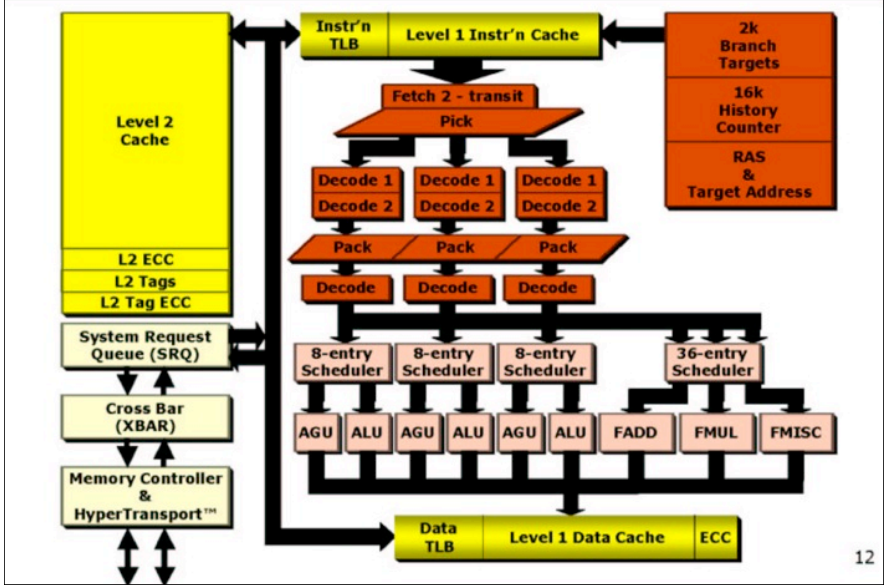
More pgprof



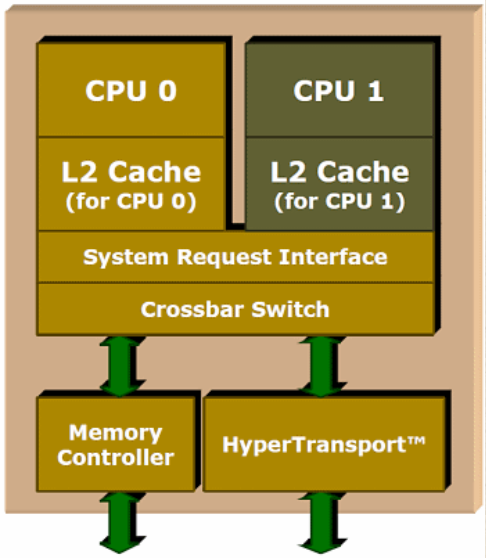
Memory Hierarchy



Processor Core Overview



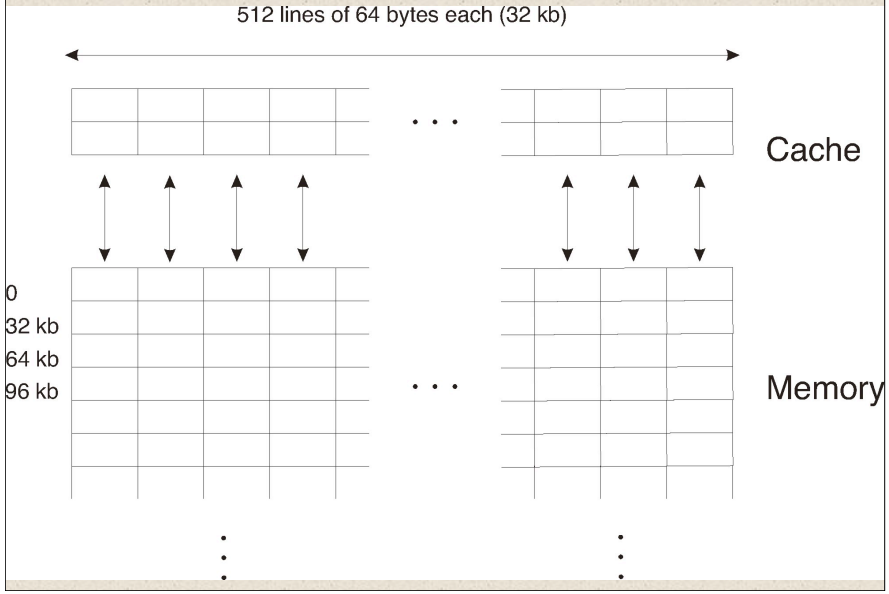
The Opteron chip



Cache organization

- Cache line
  - Smallest unit of data that can be transferred to or from memory
- Direct mapped
  - Only one possible location in cache for a particular memory location
- Set-associative
  - Multiple possible locations in cache for a given memory location

Opteron L1 data cache



AMD dual-core opteron 275

- L1 Cache
  - 2 x 64k instruction cache/processor
    - Two-way associative, 64 byte line
  - 2 x 64k data cache/processor
    - Two-way associative, 64 byte line
- L2 Cache
  - 2 x 1M
    - Sixteen-way associative, 64 byte line
- Translation Lookaside Buffer (TLB)
  - L1 Data: 40 entry fully associative, 4M/2M, 4k pages
  - L2 Data: 512 entry 4-way associative

Translation Lookaside Buffer (TLB)

- Memory is allocated by the operating system in pages of physical memory
- A program uses virtual address space which may be scattered on the physical memory
- The translation between physical memory page and virtual memory is kept in the Translation Lookaside Buffer

Cache thrashing

- Assume 16k direct mapped cache

```
real*8 x(2048,2048)
do i = 1, 2048
  do j = 2, 2048
    x(i,j) = x(i,j)+x(i,j-1)
  enddo
enddo
```

Good Coding Practices

- Access data with unit stride
- Avoid expensive operations such as divide and exponent
- Avoid If statements in loops
- Avoid subroutine and function calls in loops
- Avoid implicit type conversion
- Avoid leading array dimensions equal to a power of two
- Avoid global variables
- Avoid using pointers
- Maybe use table lookup instead of repetitive calculations
- Avoid excessive hand tuning

Optimize I/O

- Eliminate or reduce I/O
  - Older programs used scratch files to reduce memory usage
- Reduce paging
  - Use less memory or a computer with a larger memory
- Use unformatted I/O and long record lengths

Compiler Options

Optimization	Time
pgf77 -O0	56 sec.
pgf77 -O1 (default)	56 sec.
pgf77 -O2	45 sec.
pgf77 -O3	42 sec.
pgf77 -fast -fastsse	34 sec.
ifort -O3 -fp-model precise	28 sec
ifort -O3 -xW -fp-model precise	16 sec
ifort -O3 -xW	16 sec.
gfortran -O3	53 sec.
pgcc -O3	51 sec.
icc -O3 -xW -fp-model precise	60 sec.

Profile guided optimization

- -Mpfi: produces instrumented code
- Run the instrumented code
- -Mpfo: recompile using the data generated from the instrumented code
- For Intel compiler: -prof-gen, -prof-use

Loop unrolling

- Inner loop unrolling - keep FPU busy

```
DO  I=1,M
      A(I)=A(I)+P*B(I)
ENDDO


DO  I=1,M,4
      A(I)=A(I)+P*B(I)
      A(I+1)=A(I+1)+P*B(I+1)
      A(I+2)=A(I+2)+P*B(I+2)
      A(I+3)=A(I+3)+P*B(I+3)
ENDDO
```

- Outer loop unrolling - increase FMA to load/store ratio

Avoid array dimensions of powers of 2

```
parameter (ldx = 256, ldy = 256, ldz = 256)
parameter (nx  = 256, ny  = 256, nz  = 256)
real*8 data0(ldx,ldy,ldz),data1(ldx,ldy,ldz)
real*8 data2(ldx,ldy,ldz),data3(ldx,ldy,ldz)
```

```
#define LDX 256
#define LDY 256
#define LDZ 256
```

zsweep

```
do j = 1, ny
    do i = 1, nx
      do k = 1, nz
        if(k .eq. 1)then
          v1(i,j,k) = 0.0
        elseif(k .eq. nz)then
          v1(i,j,k) = 1.0
        else
          v1(i,j,k) = v0(i,j,k)+half*v0(i,j,k-1)+v0(i,j,k+1)
        endif
      enddo
    enddo
  enddo
```

zsweep-opt

```
do j = 1, ny
      do i = 1, nx
           v1(i,j,1) = 0.0
        do k = 2, nz-1
           v1(i,j,k) = v0(i,j,k)+half*v0(i,j,k-1)+v0(i,j,k+1)
        enddo
           v1(i,j,nz) = 1.0
      enddo
    enddo
```

```
for (j=0;j<ny;j++){
  for (i=0;i<nx;i++){
       v1[0][j][i] = 0.0;
    for (k=1;k<(nz-1);k++)
       v1[k][j][i] = v[k][j][i]+half*v[k-1][j][i]+v[k+1][j][i];
    v1[nz-1][j][i]=1.0;
  }
 }
```

mmul

```
do n = 1, nn
   do k = 1, nk
     do m = 1, nm
       v0(m,n,l) = v0(m,n,l) + v1(m,k,l)*v2(k,n,l)
     enddo
   enddo
 enddo
```

Scientific libraries

- blas and lapack
  - Hardware optimized versions
    - AMD: acml
    - INTEL: mkl
- fftw
  - Portable
  - Optimizes for hardware

mmul-opt

```
call dgemm('N','N',nm,nn,nk,1.d0,v1(1,1,l),
     .           ldx,v2(1,1,l),ldx,1.d0,v0(1,1,l),ldx)


pgf77 -o prob1 prob1-opt.f -lacml
```

```
#include <acml.h>
dgemm('N','N',nm,nn,nk,1.0L,&v1[l][0][0],LDX,&v2[l][0][0],LDX,1.0L,&v[l][0]
    [0],LDX);

pgcc -o prob1 prob1-opt.c -lacml -lpgftnrtl
```

```
#include <mkl_cblas.h>
cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans,nm,nn,nk,1.0L,&v1[l][0]
[0],LDX,&v2[l][0][0],LDX,1.0L,&v[l][0][0],LDX);

icc -o prob1 prob1-opt.c  -I/usr/local/intel/mkl/9.1.023/include/ -L$MKLPATH -lmkl -
lguide -lpthread -fp-model precise -Wl,-rpath,$MKLPATH
```

ysweep

```
do k = 1, nz
   do i = 1, nx
     do j = 1, ny
       v1(i,j,k) = dsin(i*(pi/nx))+dcos(v0(i,j,k))
     enddo
   enddo
 enddo
```

ysweep-opt

```
   do i = 1, nx
     sinx(i)=dsin(i*(pi/nx))
   enddo
   do k = 1, nz
     do j = 1, ny
     do i = 1, nx
       v1(i,j,k) = sinx(i)+dcos(v0(i,j,k))
     enddo
    enddo
  enddo
```

```
for (i=0;i<nx;i++)
      sinx[i]=sin((i+1)*(pi/nx));
 for (k=0;k<nz;k++){
    for (j=0;j<ny;j++){
  for (i=0;i<nx;i++)
     v1[k][j][i] = sinx[i]+cos(v[k][j][i]);
  }
 }
```

xsweep

```
do k = 1, nz
   do j = 1, ny
     do i = 1, nx
       v1(i,j,k) = v1(i,j,k)+(v0(i,j,k)**2)/17
     enddo
   enddo
 enddo
```

xsweep-opt

```
fact=1.0d0/17
     do k = 1, nz
       do j = 1, ny
         do i = 1, nx
           v1(i,j,k) = (v0(i,j,k)*v0(i,j,k))*fact
         enddo
       enddo
     enddo
```

```
factor = 1.0L/17;
 for (k=0;k<nz;k++){
   for (i=0;i<nx;i++){
     for (j=0;j<ny;j++)
       v1[k][j][i] = (v[k][j][i]*v[k][j][i])*factor;
   }
 }
```

Optimizing prob1

	original	Original, compiler optimized	Hand tuned	Hand tuned, compiler optimized
Fortran	56	16	18	17
c	57	51	18	17

Sources of information

- The Portland Group
  http://www.pgroup.com/resources/docs.htm
- Intel
  http://www.ualberta.ca/AICT/RESEARCH/LinuxClusters/doc/ifc91/Doc_Index.htm
  http://www.ualberta.ca/AICT/RESEARCH/LinuxClusters/doc/icc91/Doc_Index.htm
- AICT General Purpose Linux Cluster
  http://www.ualberta.ca/AICT/RESEARCH/LinuxClusters/