King Abdullah University of
Science and Technology

KAUST

Code profiling on Shaheen using Xprof and MPI_Trace.

Nicholas Allsopp (KAUST)

Content

Xprof: Hardware (CPU) performance

MPI_Trace: Message-passing performance

Xprof

Subroutine + line level profiling.

Using the IBM compilers, compile and link: -g –pg -qfullpath

The -g option can be used along with optimization, but sometimes the actual parent or grandparent may be off a line or two.

Run as usual.

Generates output files called gmon.out.0….n-1

Xprof a.out gmon.out.0

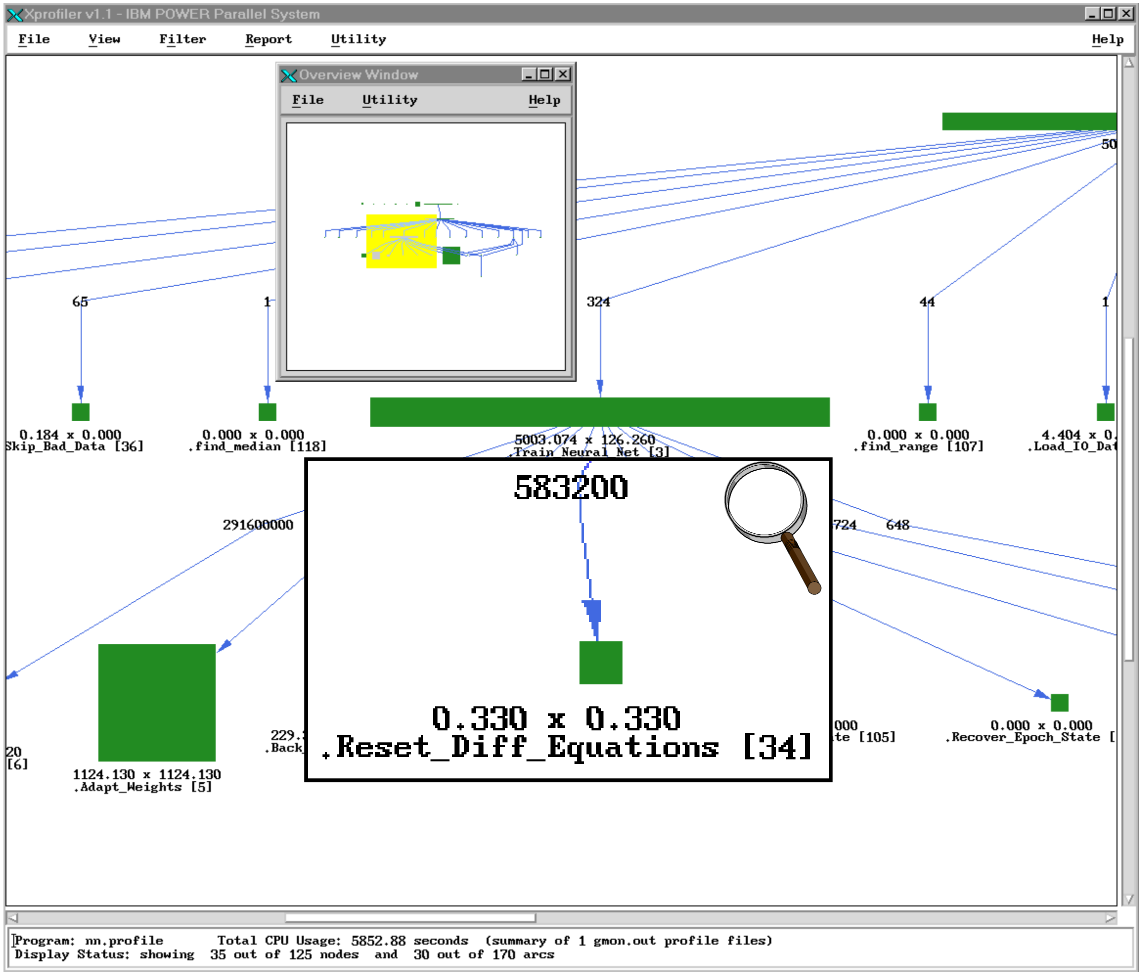
Xprof: Main Display

Width of a bar:
time including called routines

Height of a bar:
time excluding called routines

Call arrows
labeled with number of calls

Overview window
for easy navigation
(View → Overview)

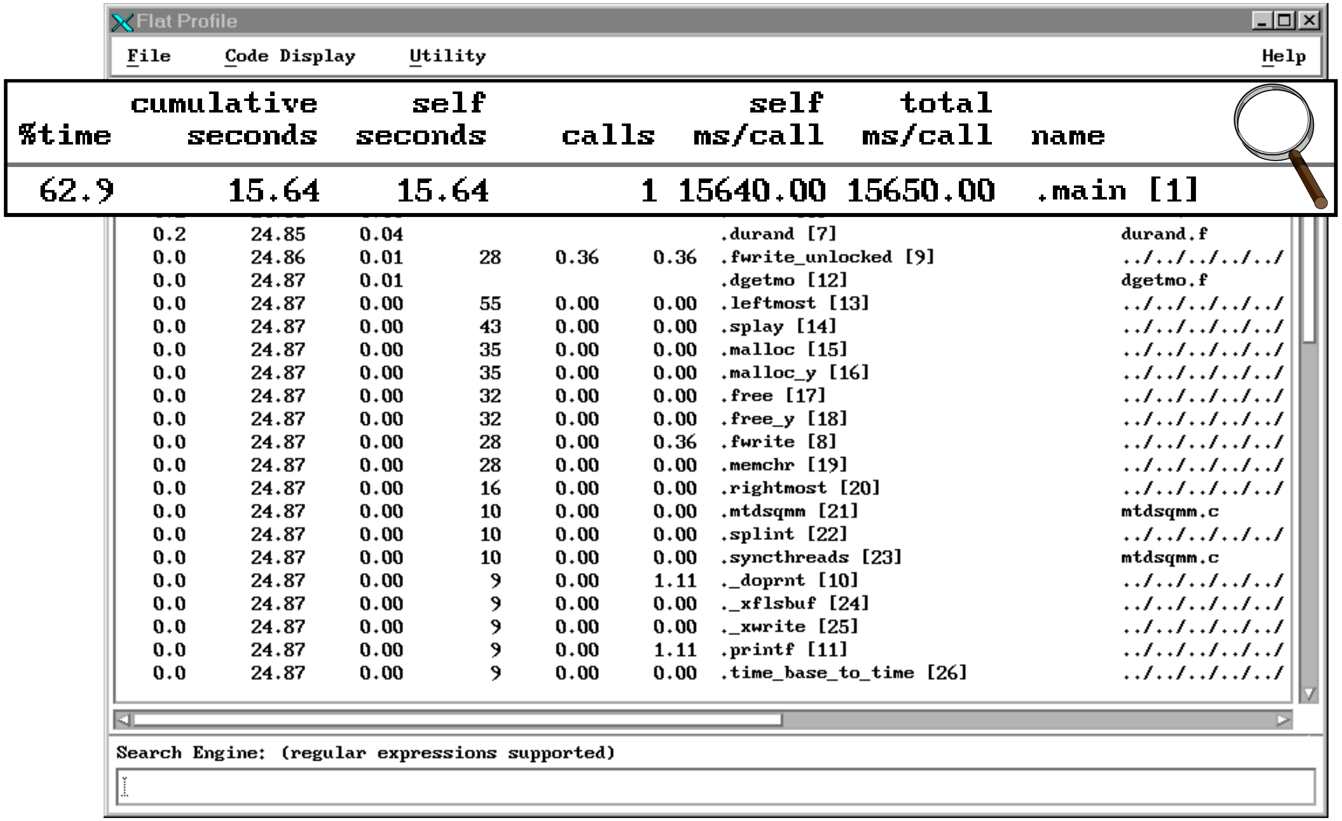


```
Program: nn.profile        Total CPU Usage: 5852.88 seconds  (summary of 1 gmon.out profile files)
Display Status: showing  35 out of 125 nodes  and  30 out of 170 arcs
```

Xprof: Flat Profile

Menu **Report** provides usual gprof reports plus some extra ones

- Flat Profile
- Call Graph Profile
- Function Index
- Function Call Summary
- Library Statistics



```
          cumulative    self              self      total
%time        seconds  seconds     calls  ms/call   ms/call   name

 62.9          15.64    15.64         1 15640.00 15650.00   .main [1]
```

```
  0.2     24.85    0.04                             .durand [7]                  durand.f
  0.0     24.86    0.01       28     0.36     0.36  .fwrite_unlocked [9]         ../../../../../
  0.0     24.87    0.01                             .dgetmo [12]                 dgetmo.f
  0.0     24.87    0.00       55     0.00     0.00  .leftmost [13]               ../../../../../
  0.0     24.87    0.00       43     0.00     0.00  .splay [14]                  ../../../../../
  0.0     24.87    0.00       35     0.00     0.00  .malloc [15]                 ../../../../../
  0.0     24.87    0.00       35     0.00     0.00  .malloc_y [16]               ../../../../../
  0.0     24.87    0.00       32     0.00     0.00  .free [17]                   ../../../../../
  0.0     24.87    0.00       32     0.00     0.00  .free_y [18]                 ../../../../../
  0.0     24.87    0.00       28     0.00     0.36  .fwrite [8]                  ../../../../../
  0.0     24.87    0.00       28     0.00     0.00  .memchr [19]                 ../../../../../
  0.0     24.87    0.00       16     0.00     0.00  .rightmost [20]              ../../../../../
  0.0     24.87    0.00       10     0.00     0.00  .mtdsqmm [21]                mtdsqmm.c
  0.0     24.87    0.00       10     0.00     0.00  .splint [22]                 ../../../../../
  0.0     24.87    0.00       10     0.00     0.00  .syncthreads [23]            mtdsqmm.c
  0.0     24.87    0.00        9     0.00     1.11  ._doprnt [10]                ../../../../../
  0.0     24.87    0.00        9     0.00     0.00  ._xflsbuf [24]               ../../../../../
  0.0     24.87    0.00        9     0.00     0.00  ._xwrite [25]                ../../../../../
  0.0     24.87    0.00        9     0.00     1.11  .printf [11]                 ../../../../../
  0.0     24.87    0.00        9     0.00     0.00  .time_base_to_time [26]      ../../../../../
```

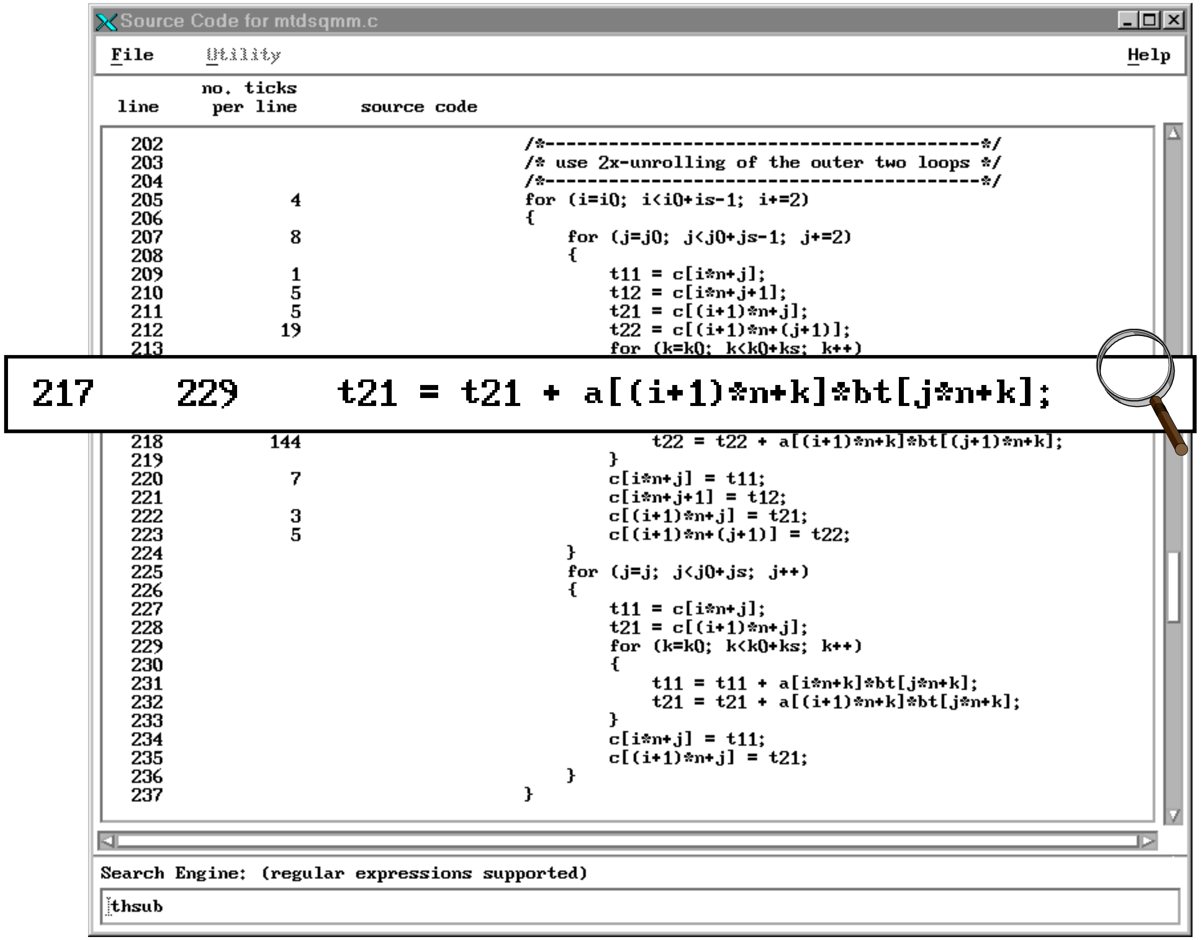
Search Engine: (regular expressions supported)

Xprof: Source Code Window

Source code window displays source code with time profile (in ticks=.01 sec)

Access

- Select function in main display
- → context menu
- Select function in flat profile
- → Code Display
- → Show Source Code

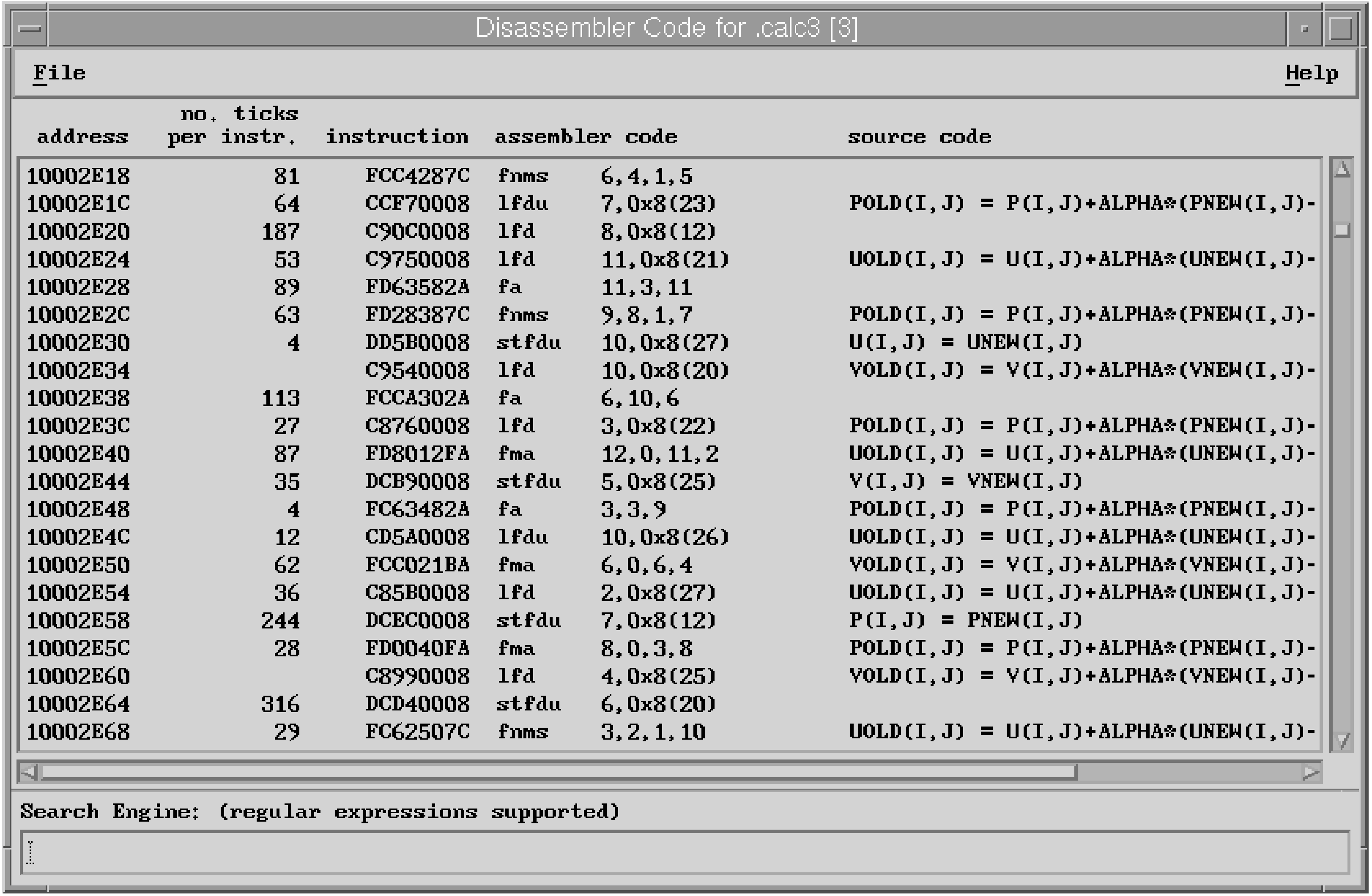


```
          no. ticks
line       per line        source code

 202                              /*----------------------------------------*/
 203                              /* use 2x-unrolling of the outer two loops */
 204                              /*----------------------------------------*/
 205           4                  for (i=i0; i<i0+is-1; i+=2)
 206                              {
 207           8                      for (j=j0; j<j0+js-1; j+=2)
 208                                  {
 209           1                          t11 = c[i*n+j];
 210           5                          t12 = c[i*n+j+1];
 211           5                          t21 = c[(i+1)*n+j];
 212          19                          t22 = c[(i+1)*n+(j+1)];
 213                                      for (k=k0; k<k0+ks; k++)

217     229     t21 = t21 + a[(i+1)*n+k]*bt[j*n+k];

 218         144                              t22 = t22 + a[(i+1)*n+k]*bt[(j+1)*n+k];
 219                                      }
 220           7                          c[i*n+j] = t11;
 221                                      c[i*n+j+1] = t12;
 222           3                          c[(i+1)*n+j] = t21;
 223           5                          c[(i+1)*n+(j+1)] = t22;
 224                                  }
 225                                  for (j=j; j<j0+js; j++)
 226                                  {
 227                                      t11 = c[i*n+j];
 228                                      t21 = c[(i+1)*n+j];
 229                                      for (k=k0; k<k0+ks; k++)
 230                                      {
 231                                          t11 = t11 + a[i*n+k]*bt[j*n+k];
 232                                          t21 = t21 + a[(i+1)*n+k]*bt[j*n+k];
 233                                      }
 234                                      c[i*n+j] = t11;
 235                                      c[(i+1)*n+j] = t21;
 236                                  }
 237                              }
```

Search Engine: (regular expressions supported)

thsub

Xprof - Disassembler Code


Disassembler Code for .calc3 [3]
File
Help
Search Engine: (regular expressions supported)

address	no. ticks per instr.	instruction	assembler code		source code
10002E18	81	FCC4287C	fnms	6,4,1,5	
10002E1C	64	CCF70008	lfdu	7,0x8(23)	POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-
10002E20	187	C90C0008	lfd	8,0x8(12)	
10002E24	53	C9750008	lfd	11,0x8(21)	UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-
10002E28	89	FD63582A	fa	11,3,11	
10002E2C	63	FD28387C	fnms	9,8,1,7	POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-
10002E30	4	DD5B0008	stfdu	10,0x8(27)	U(I,J) = UNEW(I,J)
10002E34		C9540008	lfd	10,0x8(20)	VOLD(I,J) = V(I,J)+ALPHA*(VNEW(I,J)-
10002E38	113	FCCA302A	fa	6,10,6	
10002E3C	27	C8760008	lfd	3,0x8(22)	POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-
10002E40	87	FD8012FA	fma	12,0,11,2	UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-
10002E44	35	DCB90008	stfdu	5,0x8(25)	V(I,J) = VNEW(I,J)
10002E48	4	FC63482A	fa	3,3,9	POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-
10002E4C	12	CD5A0008	lfdu	10,0x8(26)	UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-
10002E50	62	FCC021BA	fma	6,0,6,4	VOLD(I,J) = V(I,J)+ALPHA*(VNEW(I,J)-
10002E54	36	C85B0008	lfd	2,0x8(27)	UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-
10002E58	244	DCEC0008	stfdu	7,0x8(12)	P(I,J) = PNEW(I,J)
10002E5C	28	FD0040FA	fma	8,0,3,8	POLD(I,J) = P(I,J)+ALPHA*(PNEW(I,J)-
10002E60		C8990008	lfd	4,0x8(25)	VOLD(I,J) = V(I,J)+ALPHA*(VNEW(I,J)-
10002E64	316	DCD40008	stfdu	6,0x8(20)	
10002E68	29	FC62507C	fnms	3,2,1,10	UOLD(I,J) = U(I,J)+ALPHA*(UNEW(I,J)-

Message-Passing Performance: MP_Profiler Library

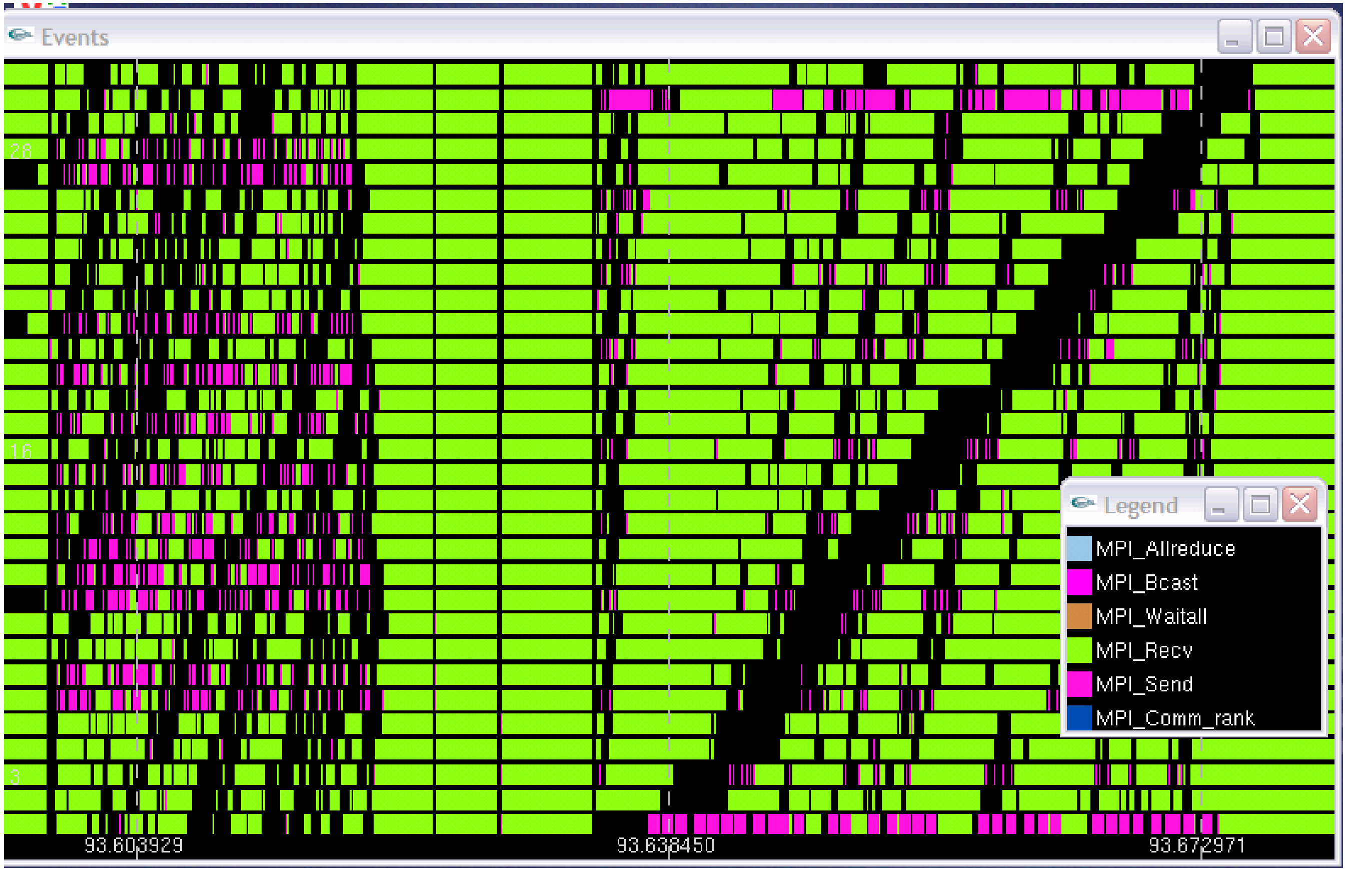
MP_Profiler

- Captures “summary” data for MPI calls with source code traceback
- No changes to source code, but MUST compile with -g
- ~1.7 microsecond overhead per MPI call
- Required link of mpitrace

Module load mpi_profile

Link in Environment variable: KSL_MPITRACE_LIB

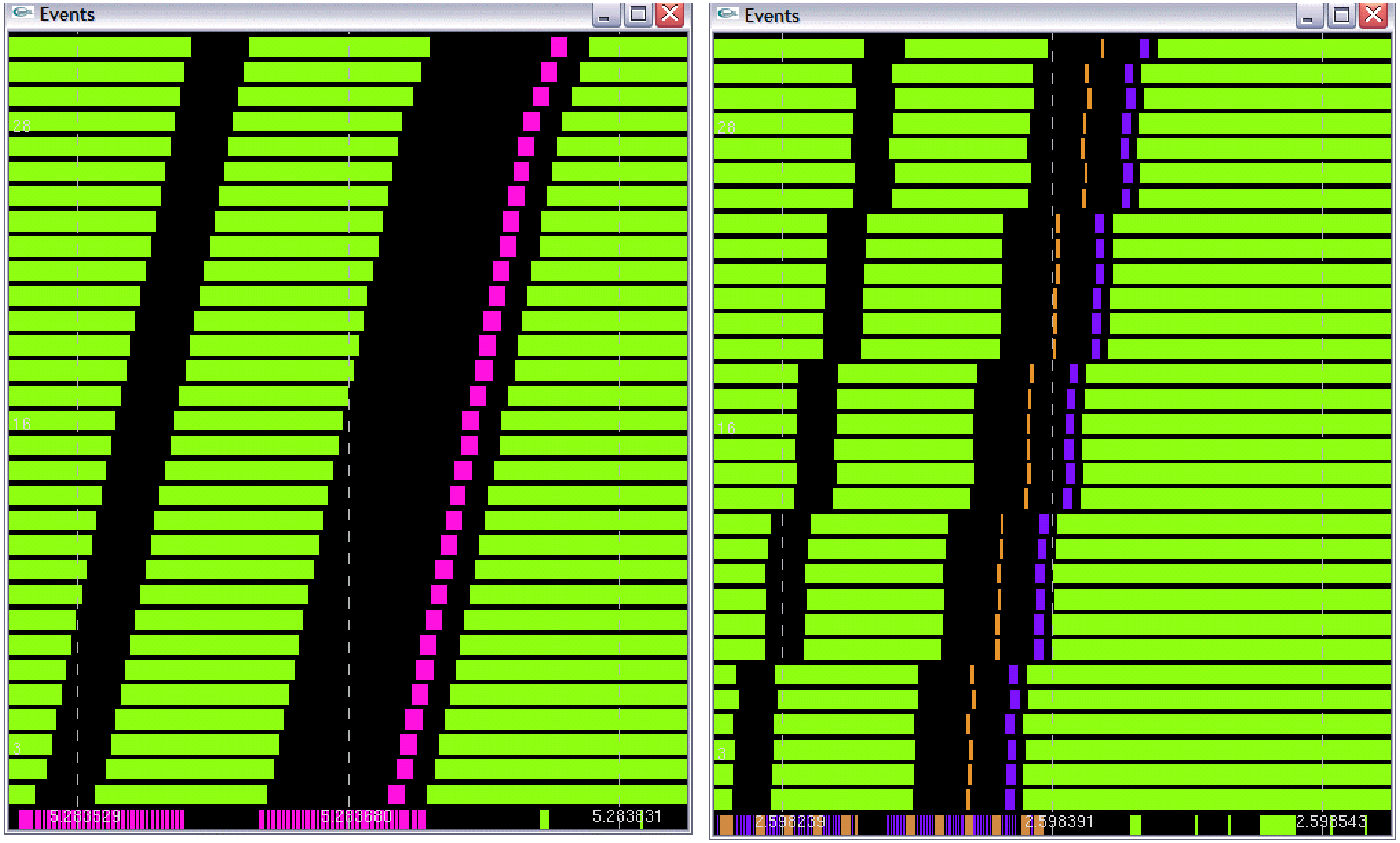
Example of domain decomposition with single master I/O



Swapping halo cells

Single node file output

Single node serialisation



Using master to groups of workers. Time is along the x-axis and each row is a different MPI task. The master node is the bottom row

Remember when Using Xprof / MPI_TRACE

Using the IBM compilers, compile and link: -g –pg –qfullpath

Module load mpi_profile

- **Link in Environment variable: KSL_MPITRACE_LIB**

Optional run-time environment variables:

- **-env "TRACE_ALL_EVENTS=yes TRACE_ALL_TASKS=yes SAVE_ALL_TASKS=yes”**

To visualise:

- **Xprof a.out gmon.out.0**
- **traceview.x events.trc**

addr2line -e a.out hex_instruction_address

Questions