Performance Degradation in the Presence of Subnormal Floating-Point Values

Orion Lawlor, Hari Govind, Isaac Dooley,
Michael Breitenfeld, Laxmikant Kale
Department of Computer Science
University of Illinois at Urbana-Champaign

http://charm.cs.uiuc.edu

Overview

- Subnormal values make computations slow.
- Serial programs are affected.
- Parallel programs are affected.
- Parallel programs may exhibit amplified slowdowns due to load imbalances between processors.
- How do I detect and fix the problem in Serial?
- How do I fix the problem in parallel?

Denormalized or Subnormal Floating Point Values

- IEEE754 Standard specifies a class of values with small magnitudes: 2^{-1074} to 2^{-1022}.
- Subnormals contain the smallest possible exponent, and mantissa with a leading zero.
- A loss of precision may occur when subnormals are used in operations.
- Programs should be notified when these operations occur.
- The worst case implementation involves expensive software traps to the OS on each operation.
- It has been claimed that occurrences of subnormals is rare and thus of no concern.

Subnormals are Bad

"Underflows occur infrequently, but when they do occur, they often come in convoys. Whatever causes one underflow will usually cause a lot more. So occasionally a program will encounter a large batch of underflows, which makes it slow."

W. Kahan

IEEE 754R minutes from September 19, 2002

Serial Example*

Jacobi(Stencil), Relaxation, and Gauss-Seidel type methods can give rise to many subnormals.

```
// initialize array
for (i = 0; i<SIZE; i++) a[i] = 0.0;
a[0] = 1.0;

// perform iterative averaging
for (j=0; j< ITER; j++)
    for (i = 2; i<SIZE-1; i++)
            a[i] = (a[i]+a[i-1]+a[i-2])*(1.0/3.0);
```

*Code available at http://charm.cs.uiuc.edu/subnormal/

Serial Example*

Jacobi(Stencil), Relaxation, and Gauss-Seidel type methods can give rise to many subnormals.

```
// initialize array
for (i = 0; i<SIZE; i++) a[i] = 10E-50;
a[0] = 1.0;

// perform iterative averaging
for (j=0; j< ITER; j++)
    for (i = 2; i<SIZE-1; i++)
            a[i] = (a[i]+a[i-1]+a[i-2])*(1.0/3.0);
```

*Code available at http://charm.cs.uiuc.edu/subnormal/

Serial Test : Worst Case Slowdown

Processor	OS	Slowdown
PowerPC G4(in an iBook)	Darwin / OSX	1.4
PowerPC G4(PowerMac)	Linux	1.6
PowerPC G4(PowerMac)	Darwin / OSX	1.7
IBM Power4	AIX 5.1	2.1
PowerPC 970 (Apple G5)	Darwin / OSX	2.2

Serial Test : Worst Case Slowdown

Processor	OS	Slowdown
PowerPC G4(in an iBook)	Darwin / OSX	1.4
PowerPC G4(PowerMac)	Linux	1.6
PowerPC G4(PowerMac)	Darwin / OSX	1.7
IBM Power4	AIX 5.1	2.1
PowerPC 970 (Apple G5)	Darwin / OSX	2.2
AMD Athlon	Linux	6.0
AMD AthlonXP	Windows XP	7.1
AMD Athlon64	Linux	21.4
AMD Opteron64	Linux	23.8

Serial Test : Worst Case Slowdown

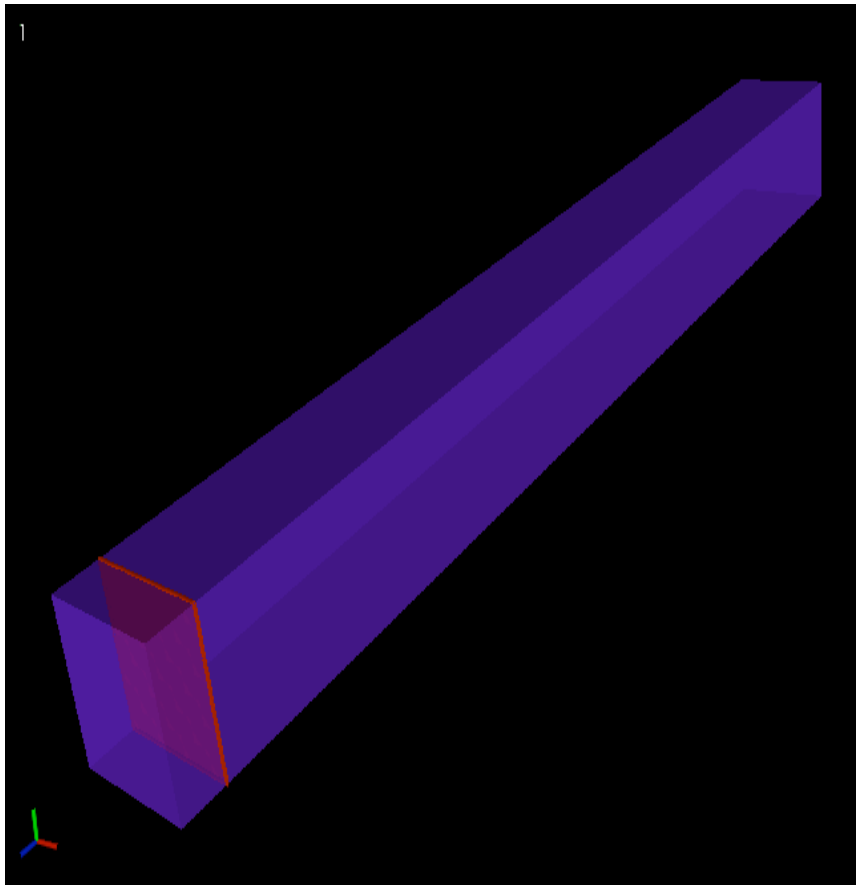
Processor	OS	Slowdown
PowerPC G4(in an iBook)	Darwin / OSX	1.4
PowerPC G4(PowerMac)	Linux	1.6
PowerPC G4(PowerMac)	Darwin / OSX	1.7
IBM Power4	AIX 5.1	2.1
PowerPC 970 (Apple G5)	Darwin / OSX	2.2
AMD Athlon	Linux	6.0
AMD AthlonXP	Windows XP	7.1
Intel Pentium 3	Linux	15.8
AMD Athlon64	Linux	21.4
AMD Opteron64	Linux	23.8
Intel Pentium 4	Linux	92.2
Intel P4 Xeon	Linux	98.2
Itanium 2 (IA-64)	Linux	183.2

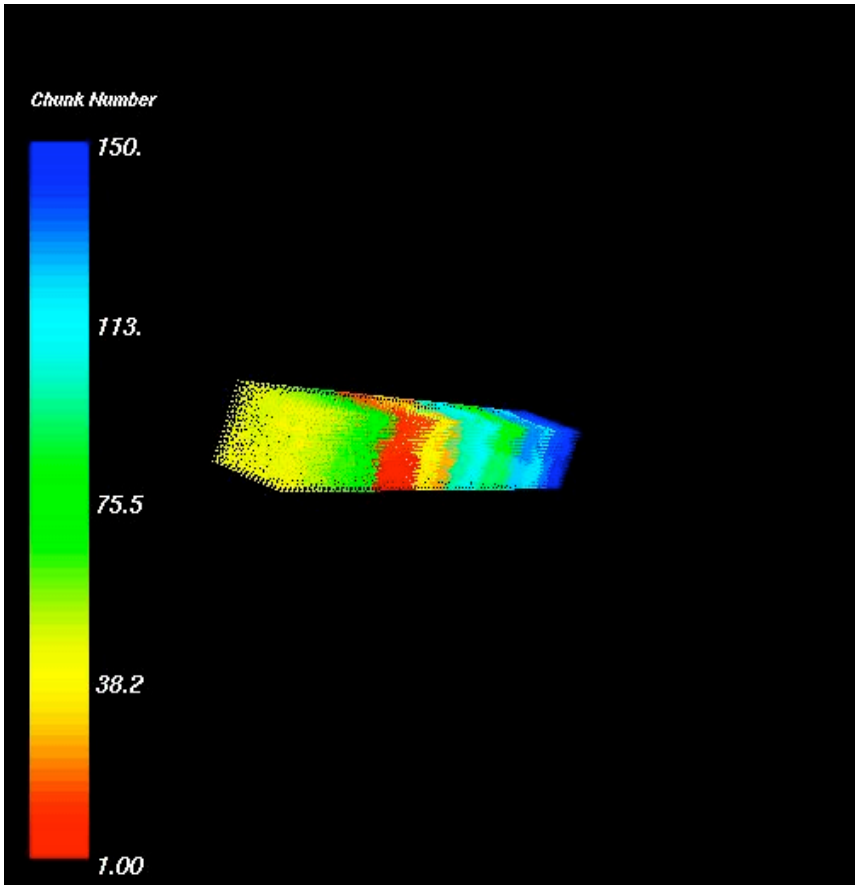
Serial Test : Worst Case Slowdown

Processor	OS	Slowdown
PowerPC G4(in an iBook)	Darwin / OSX	1.4
PowerPC G4(PowerMac)	Linux	1.6
PowerPC G4(PowerMac)	Darwin / OSX	1.7
IBM Power4	AIX 5.1	2.1
PowerPC 970 (Apple G5)	Darwin / OSX	2.2
AMD Athlon	Linux	6.0
AMD AthlonXP	Windows XP	7.1
Intel Pentium 3	Linux	15.8
AMD Athlon64	Linux	21.4
AMD Opteron64	Linux	23.8
Intel Pentium 4	Linux	92.2
Alpha EV6.8CB	Tru64 Unix	95.1
Intel P4 Xeon	Linux	98.2
Itanium 2 (IA-64)	Linux	183.2
UltraSPARC IV 64-bit	Solaris	520.0

Parallel Program where this phenomenon was observed.

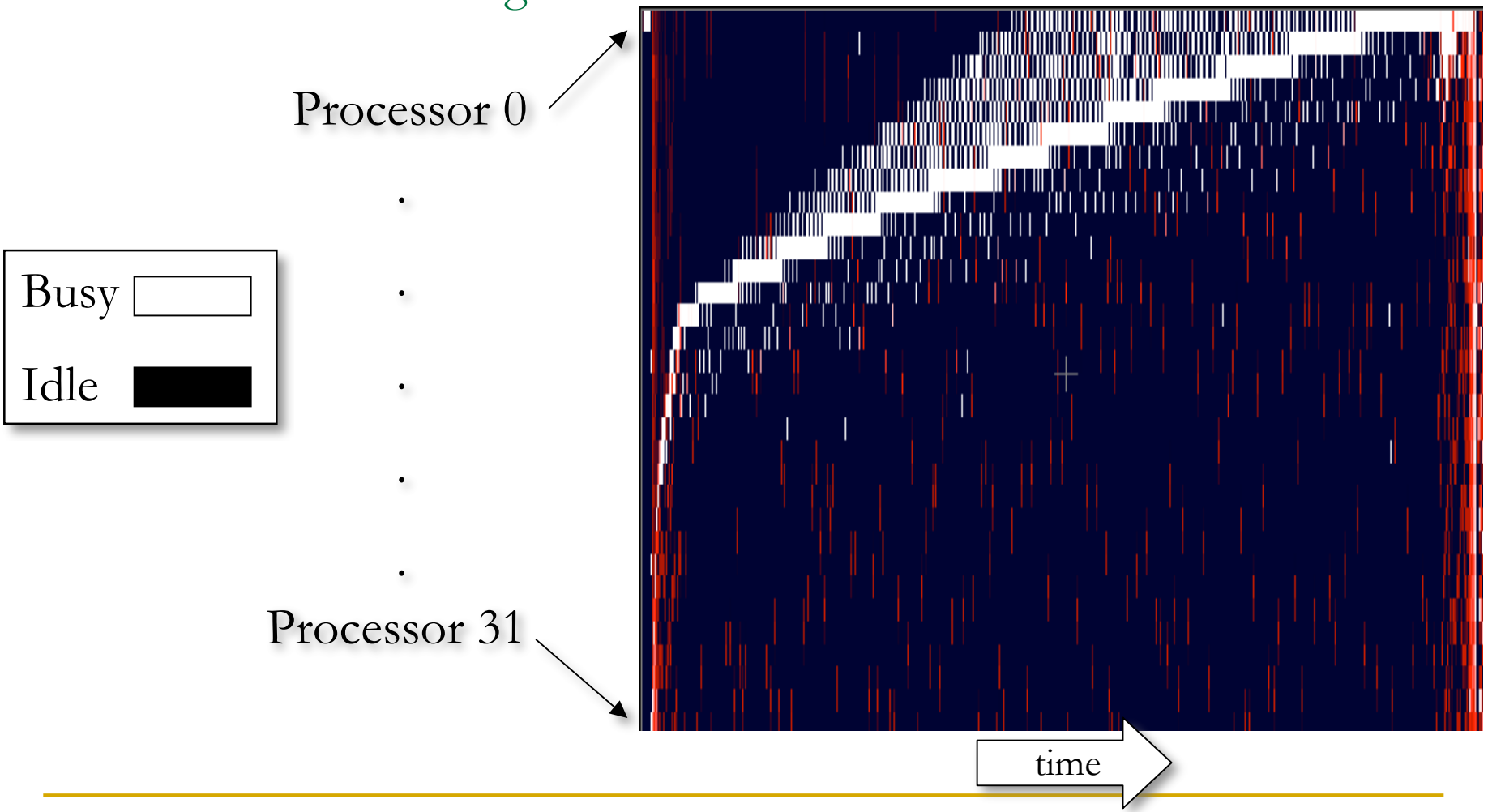
Simulating a *1D* wave propagating through a finite *3D* bar in parallel. The wavefront (left) and partitioning (right).





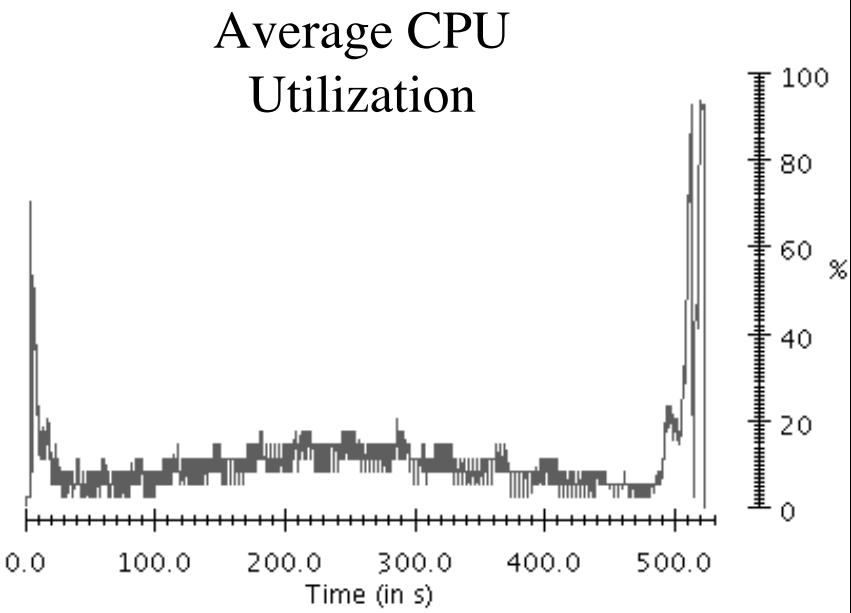
Parallel Program Timeline Overview:

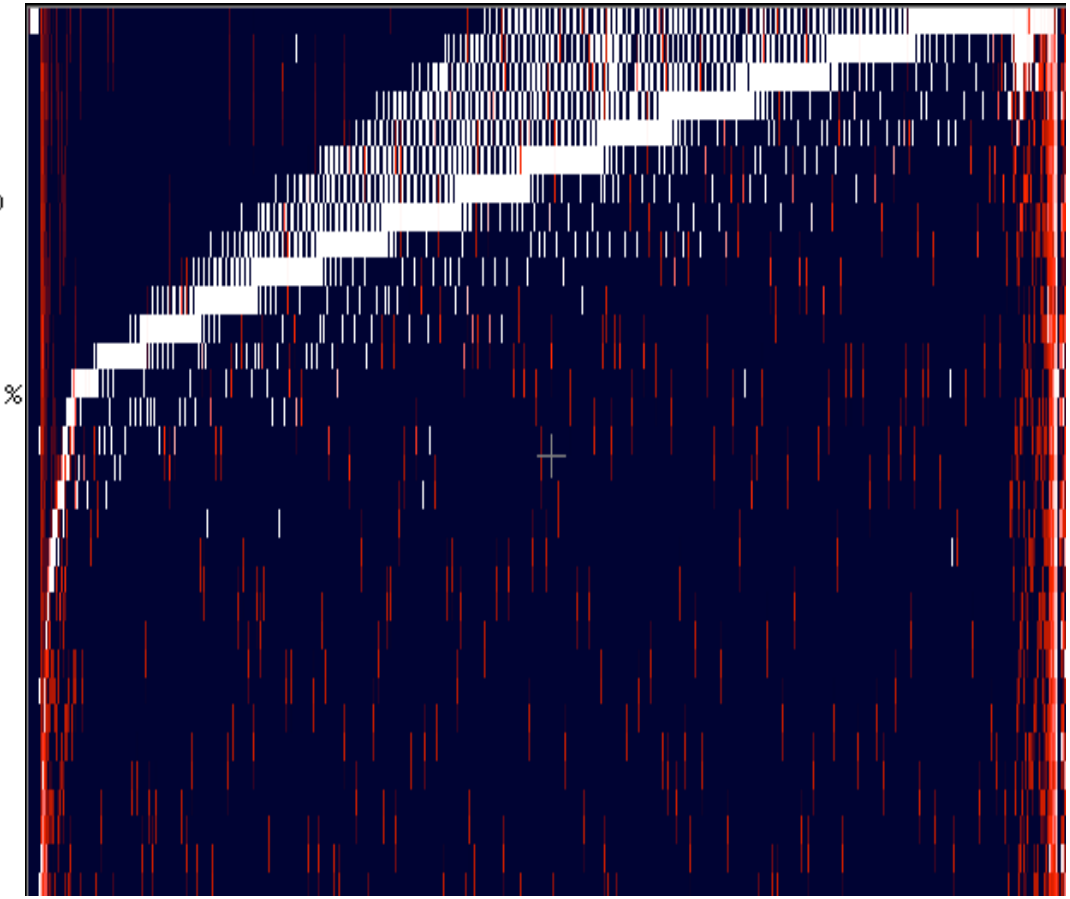
32 Alpha processors(PSC Lemieux Cluster) with Linear Partitioning



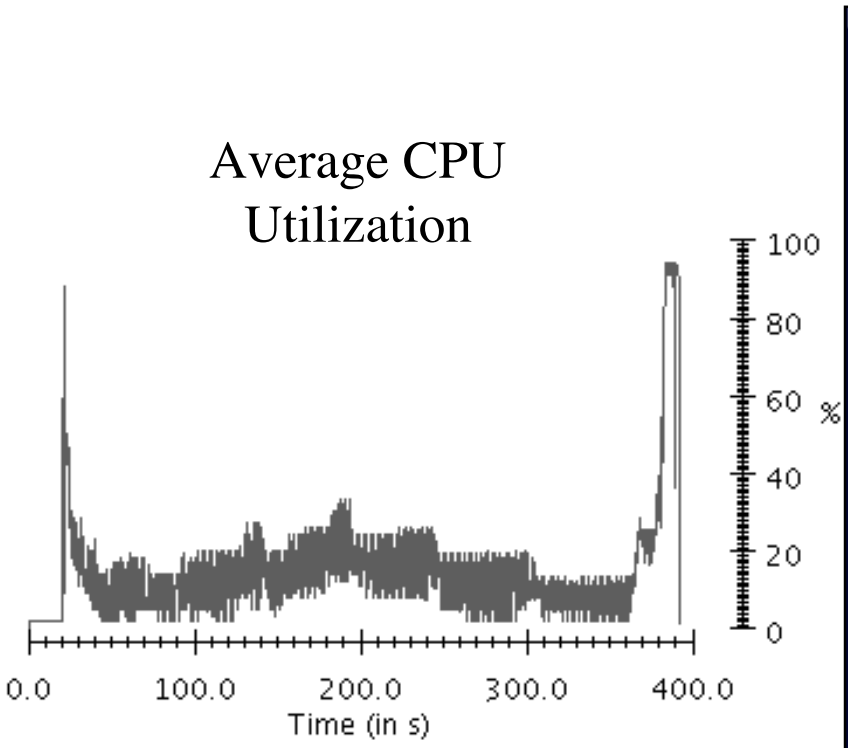
Parallel Program:

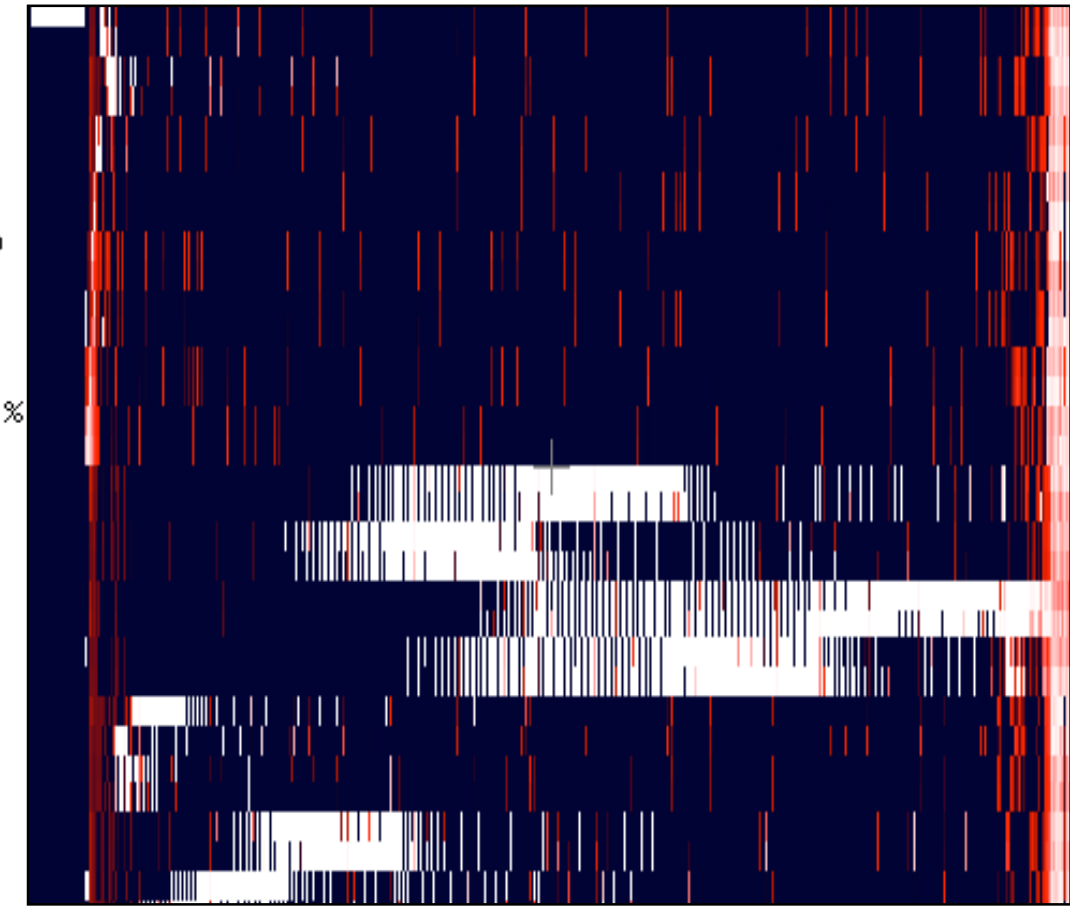
32 Alpha processors with Linear Partitions and Processor Mapping





Parallel Program:
32 Alpha processors with Normal METIS Partitioner





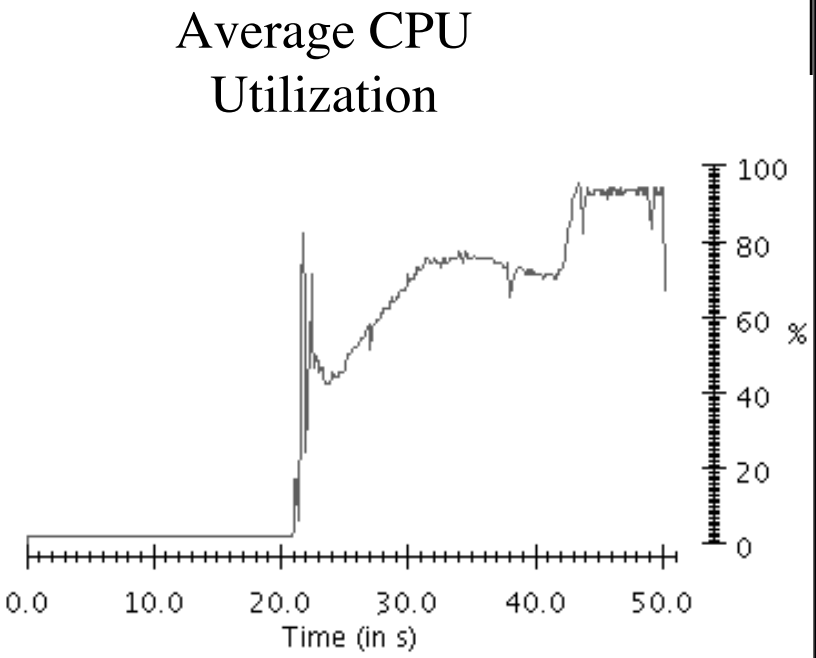
Flushing Subnormals to Zero

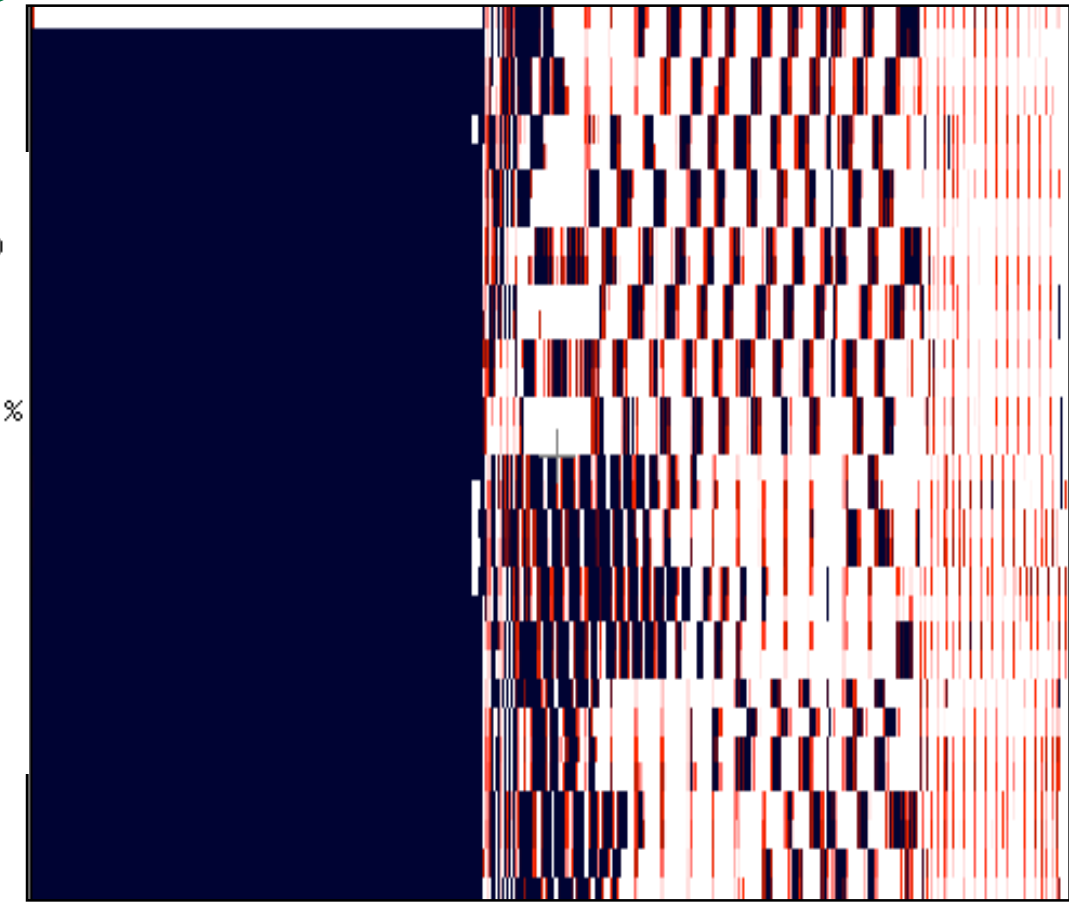
- Some processors have modes that flush any subnormal to zero.
- No subnormal will ever be produced by any floating-point operation.
- Available on some processors.
- Can be enabled by:
  - Compiler flags
  - Explicit code

Parallel Program:

32 Alpha processors with Normal METIS Partitioner

Flush Subnormals to Zero

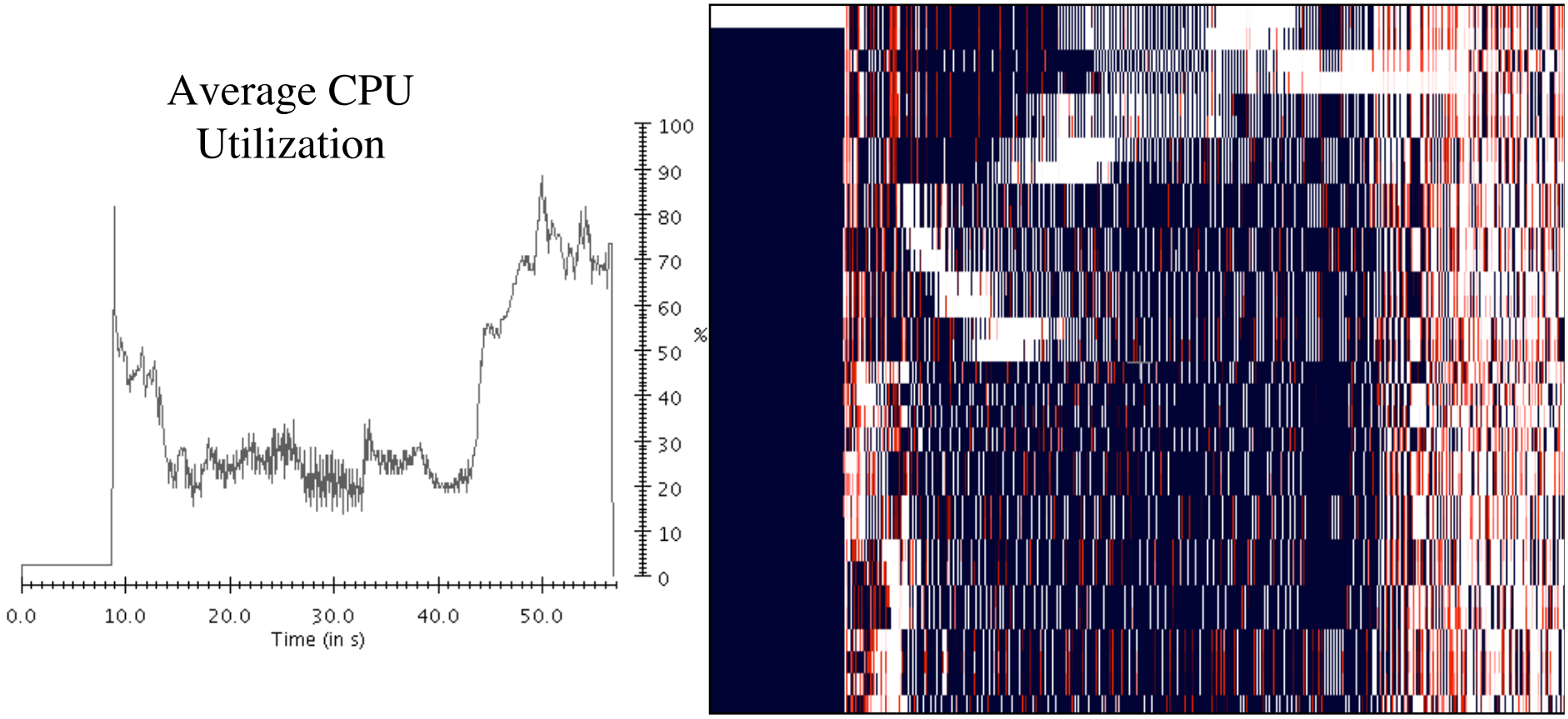




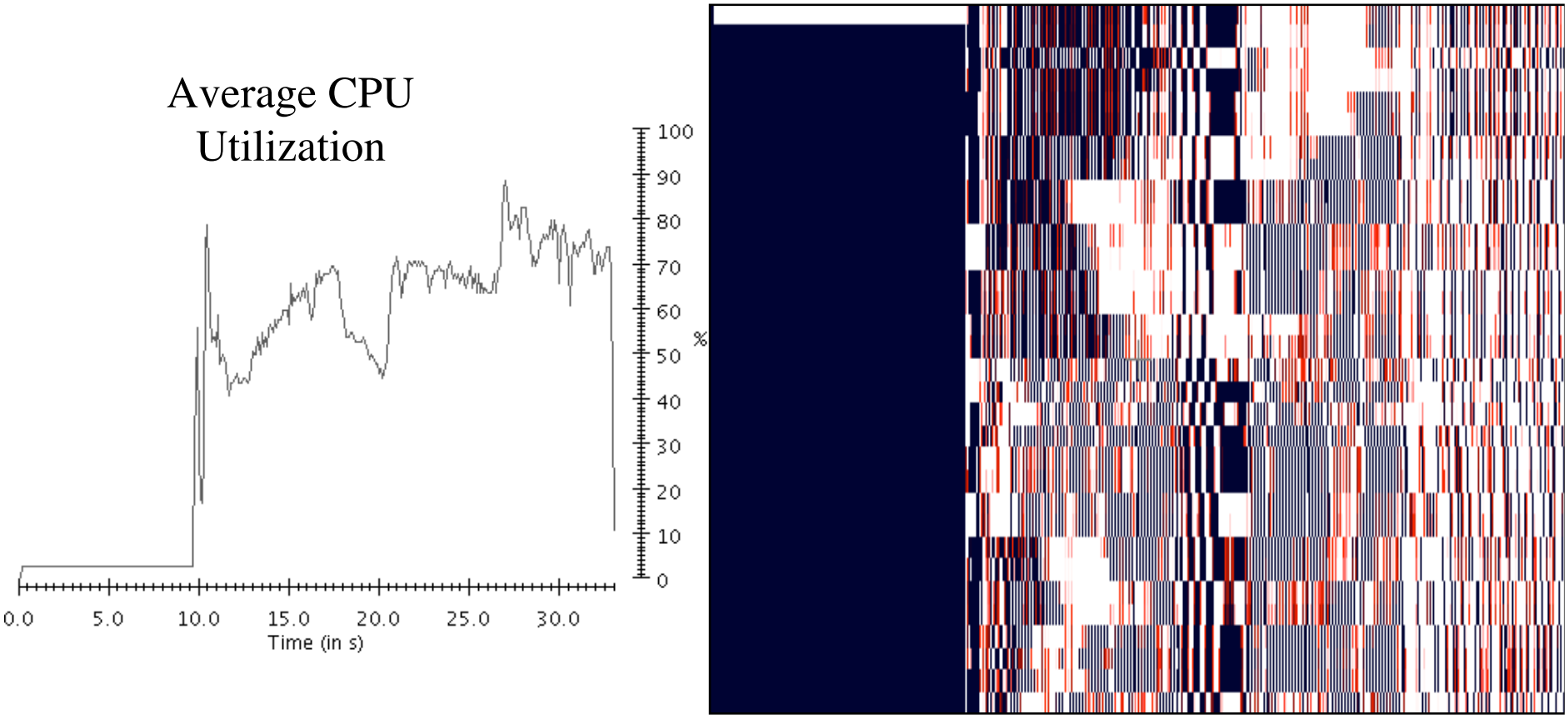
Summary of execution times on Alpha cluster

Processor mode	Execution time
Metis Partitioning with Flush To Zero (Subnormals Disabled by default)	50.3s
Metis Partitioning (Subnormals Enabled "-fpe1")	392.4s
Linear Partitioning (Subnormals Enabled "-fpe1")	522.6s

Parallel Program:
32 Xeon processors (NCSA Tungsten Cluster)



Parallel Program:
32 Xeon processors, Flush Subnormals to Zero



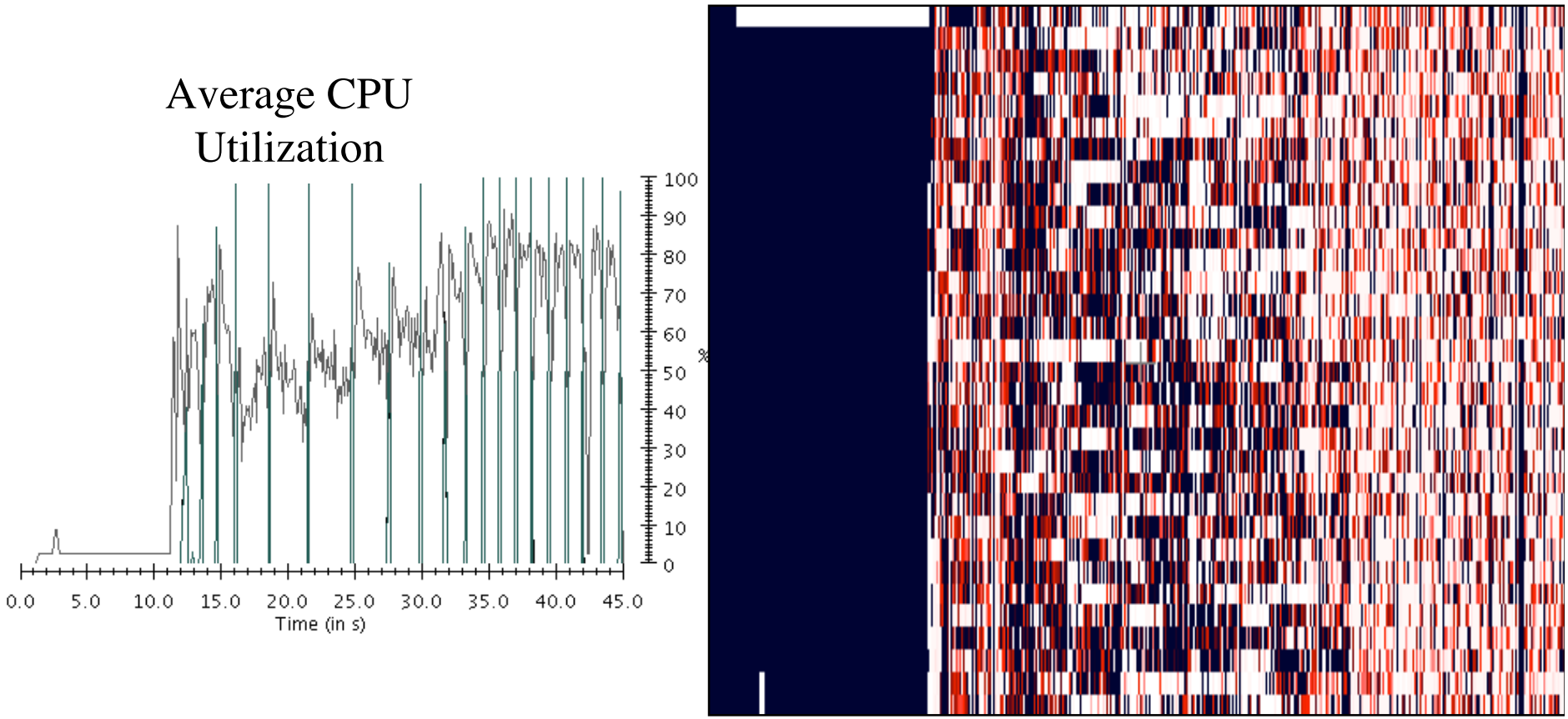
The Anatomy of The Parallel Problem

- Each processor can only proceed one step ahead of its neighbors. If the program uses any global synchronization, it will be significantly worse than the asynchronous.
- If one processor is slow, it will interfere with the progress of the rest of the processors.
- Essentially a load imbalance occurs between the processors.
- Systems built with message driven execution, virtualization / migratable objects, or loadbalancing may fix the parallel interference.

ParFUM: A Parallel Framework for Unstructured Mesh Applications

- Makes **parallelizing** a serial code faster and easier
  - Handles mesh partitioning and communication
  - **Load balancing** (via Charm++)
  - Parallel Incremental Mesh Modification:
    - Adaptivity
    - Refinement
  - Visualization Support
  - Collision Detection Library
  - High Processor Utilization even for Dynamic Physics codes

Instrument Based Load Balancing on Xeon Cluster



Parallel Program on Xeon Cluster

Processor mode	Execution time(speedup)
Default	56.8s
Flush to Zero mode	33.1s (42%)
Load Balanced	45.6s (20%)

How do I detect subnormals?

- Some compilers give an option which generates code that counts all occurrences of subnormals. E.g. DEC's "-fpe1" option
- Many compilers have options to generate IEEE compliant code. Try using such a flag and compare the results to a version with a non-IEEE compliant version.
- Explicitly scan through your data and count number of denormals. This method may yield false positives.
- It may be possible to monitor OS traps to determine if any floating-point exception handling code is executing.

How do I elminate these slowdowns?

- Use a computer system where the operations are less impacted by subnormals.
- Scale/translate data to different range if possible.
- Use a compiler option or additional code to put the processor in a mode where subnormals are flushed to zero(FTZ).
- If your numerical code cannot tolerate FTZ, then in parallel use some type of load balancing to distribute the subnormals across all processors.

Conclusion

- Subnormals are increasingly slower on modern processors.
- Parallel applications will have greatly amplified slowdowns due to the interference caused by one processor's slowdown.
- These factors may no longer be as negligible as previously thought.
- We believe there are probably many other applications with similar slowdowns.

Thank You

Parallel Programming Lab

at University of Illinois

http://charm.cs.uiuc.edu

Additional Resources and Results:

http://charm.cs.uiuc.edu/subnormal/