Performance evaluation of ASCI benchmark AZTEC using Paradyn

Final Report

Instructor:

Dr. Frank Mueller (mueller@cs.ncsu.edu)

Team Members:

Raj kumar Nagarajan (rknagara@cs.ncsu.edu)

Vikram S Poojary (vspoojar@cs.ncsu.edu)

Problem Description:
The main objective of this project is to use Paradyn, a parallel performance tool, to analyze and evaluate the performance of ASCI benchmark AZTEC. And to identify the bottlenecks in the benchmark application, to improve the performance by algorithmic changes or MPI specific changes, suggest improvements to the tool based on experience with benchmark evaluation.

In this document we discuss the following:

- AZTEC Benchmark
- Performance of AZTEC on the cluster
- Paradyn
- Evaluation of AZTEC using Paradyn
- Improvements to tool
- Location of src and binary files on cluster

AZTEC:
AZTEC is a massively parallel iterative solver library for solving sparse linear systems. It provides state-of-the-art iterative methods that performs well on parallel computers (applications of over 200 Gflops have been achieved on the Sandia-Intel TFlop Computer) and at the same time is easy to use for application engineers.

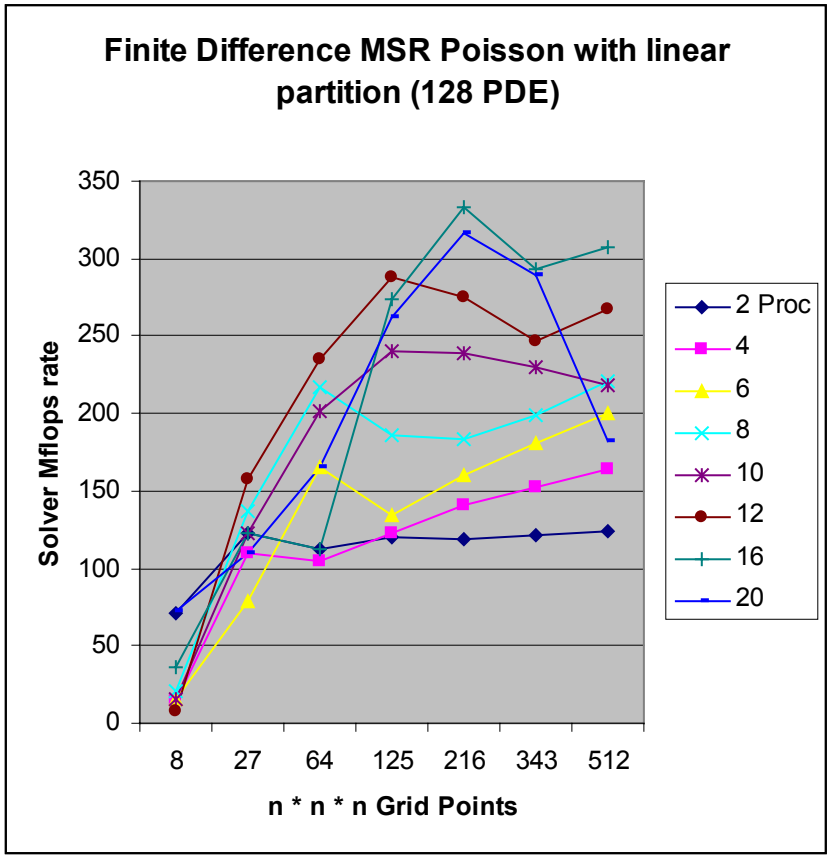
Simplicity is attained using the notion of a global distributed matrix. The global distributed matrix allows a user to specify pieces (different rows for different processors) of his application matrix exactly as he would in the serial setting (i.e. using a global numbering scheme). Issues such as local numbering, ghost variables and messages are ignored by the user and are instead computed by an automated transformation function (AZ_transform). Efficiency is achieved using standard distributed memory techniques; locally numbered submatrices, ghost variables, and message information computed by the transformation function are maintained by each processor so that local calculations and communication of data dependencies is fast. Additionally, AZTEC takes advantage of advanced partitioning techniques (Linear and Box partitioning) and utilizes efficient dense matrix algorithms when solving block sparse matrices.

Performance of AZTEC on the Cluster:

The test program solves Poisson's Equation, using Finite Difference on an n * n * n Grid. The Grid Sparse matrices are represented using DMSR (Distributed Modified Sparse Row) format, which is a slightly

modified format of MSR. In the test experiment, the cluster had 13 active processors running.

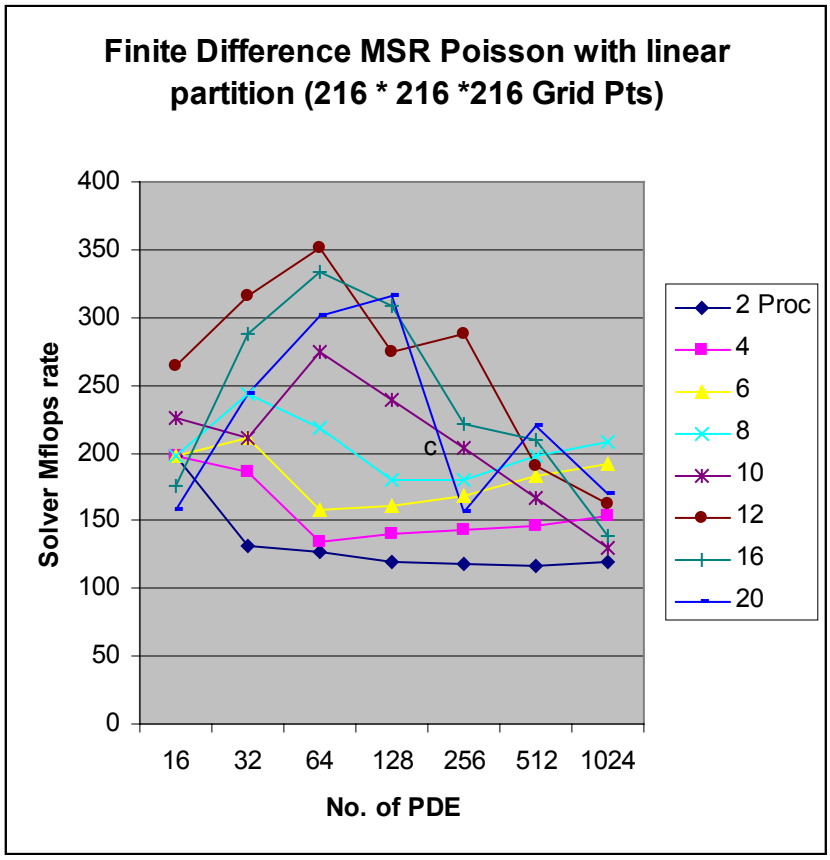
Test with varying Grid size and with 128 PDE's and Linear partition of matrix:



Observation:

1. Performace increases with increase in the number of processors.
2. But the performance incease is small as the Grid size increases. This is because of the reduction in the parallel computation on the cluster. Each node has to compute a large Grid Size.
3. For number of processors >= 20, the performance comes down. This is because of the inherent limitation of the cluster size.

Test with varying PDE's and with a GRID size of 216 and linear partition of matrix:



Test with varying Grid size and with 128 PDE's and Box partition of matrix:

Grid Size	8 Proc
8	71.66
64	281.9
216	324.5
512	316.88

Observation:

1. The Box partition gives better performance than linear partition. But it is severly limited by the number of Grid size patterns and Processors that can be used. The Grid Size needs to be a cube-root of some integer (box size across each dimension) and number of processors should be an multiple of grid size.

Paradyn:

It is a tool for measuring the performance of parallel distributed programs. It achieves this by dynamically inserting (attaching) the instrumentation code to unmodified executable. The instrumentation is

done automatically by the Performance Consultant module, which identifies the performance problems, decides where and when to collect data. It uses a W3 (Why, Where and When) Search model. The tool also provides an open interface for program visualization and can be configured for application specific performance data.

Evaluation of AZTEC using Paradyn:

The AZTEC benchmark program was run with the following parameters under Paradyn:

Num of processors: 5
Finite Difference MSR Poisson method
Grid Size: 27000
Num of PDE: 256
Linear partition

For detailed instructions to run the benchmark under Paradyn, refer to the readme file.

The following are the screenshots of the program execution:

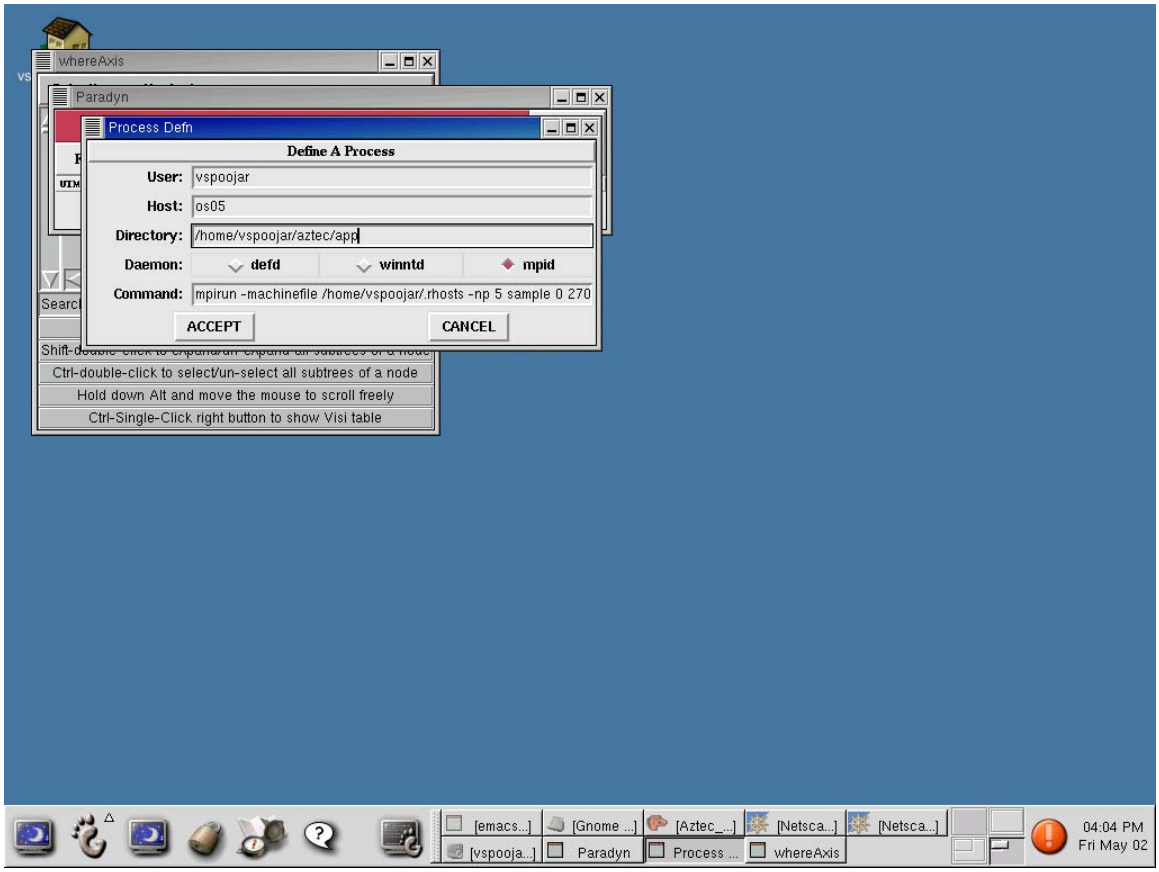


Fig 1:

The above screenshot depicts the process of launching the benchmark under paradyn.

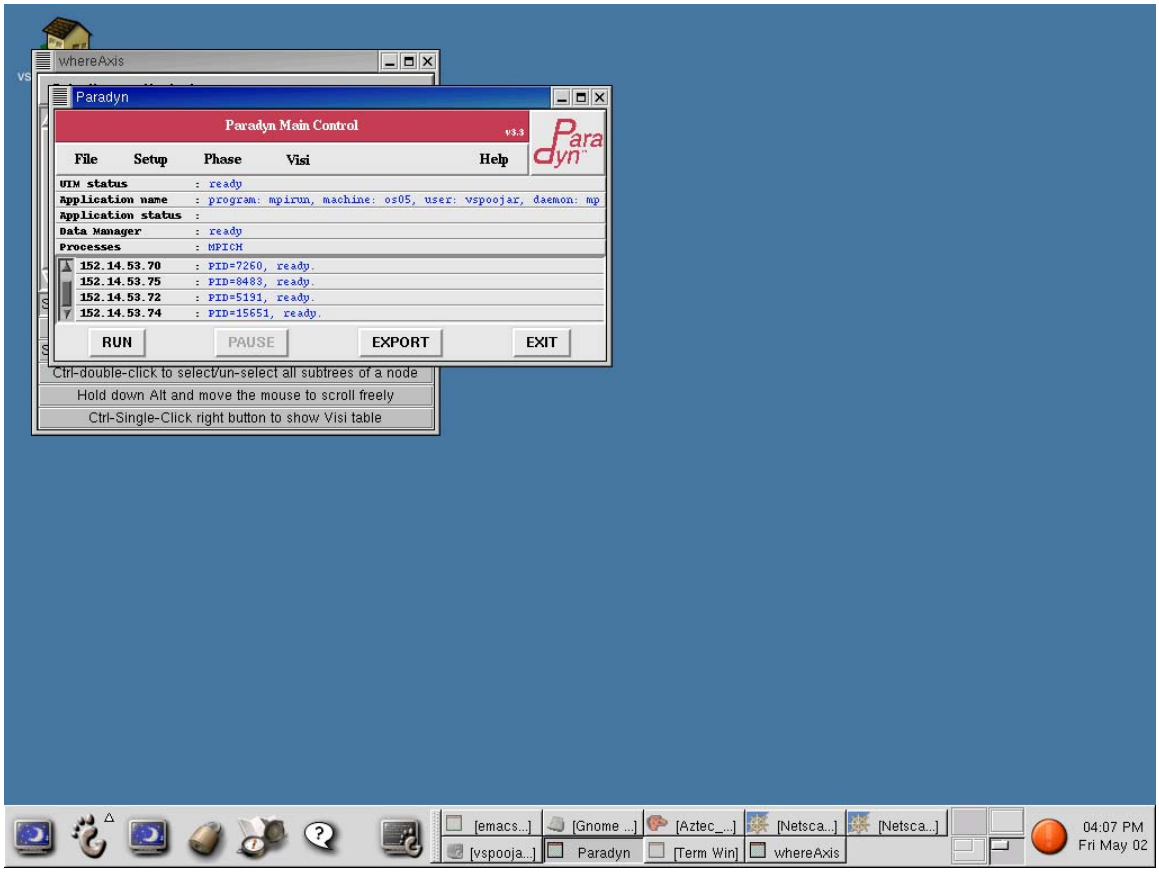


Fig 2: Application Initialized and ready to execute

The above screenshot depicts the stage when the paradynd daemons have been launched on all the nodes to monitor the MPI processes, the daemons have attached to the processes and before the steps for gathering performance metrics have been carried out.

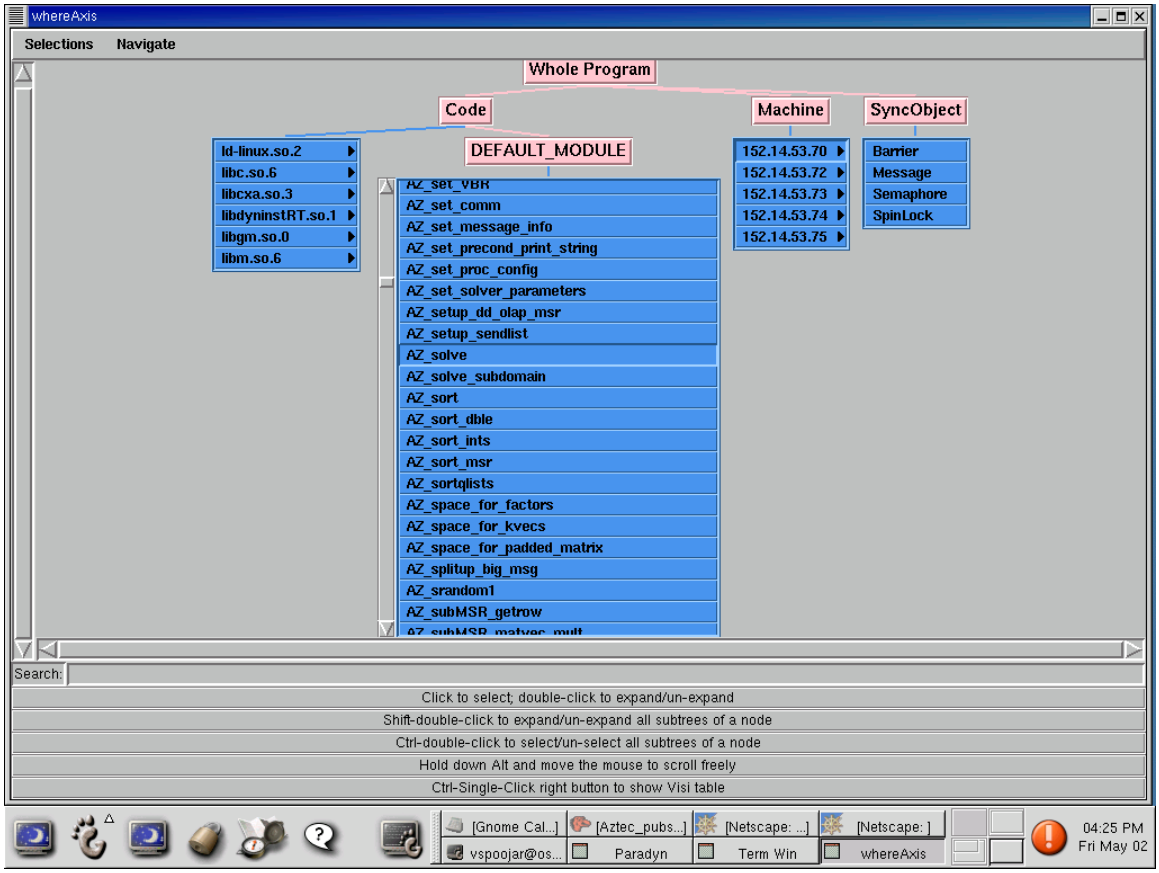


Fig 3: Components of the Program (Machine and Program functions)

The above screenshot details the where axis of the search space in W3 model of paradyn. In this screenshot, we have enabled node 152.14.53.70 and the function AZ_solve as the specific foci for which we want to gather metrics.

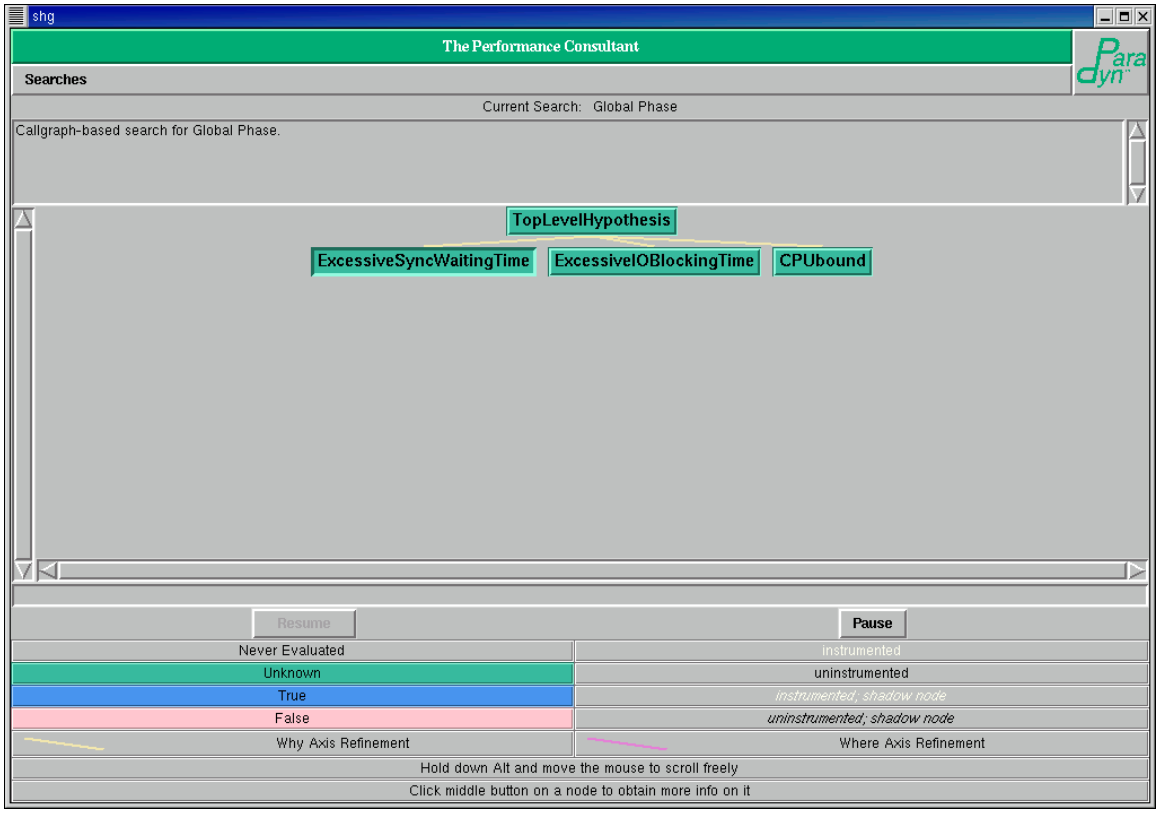


Fig 4:Initial Performance Consultant – before application run

The above screenshot is of launching the performance consultant window just before running the benchmark under paradyn. We can see that the initial hypotheses have not been tested by means of the green color on the hypothesis nodes.

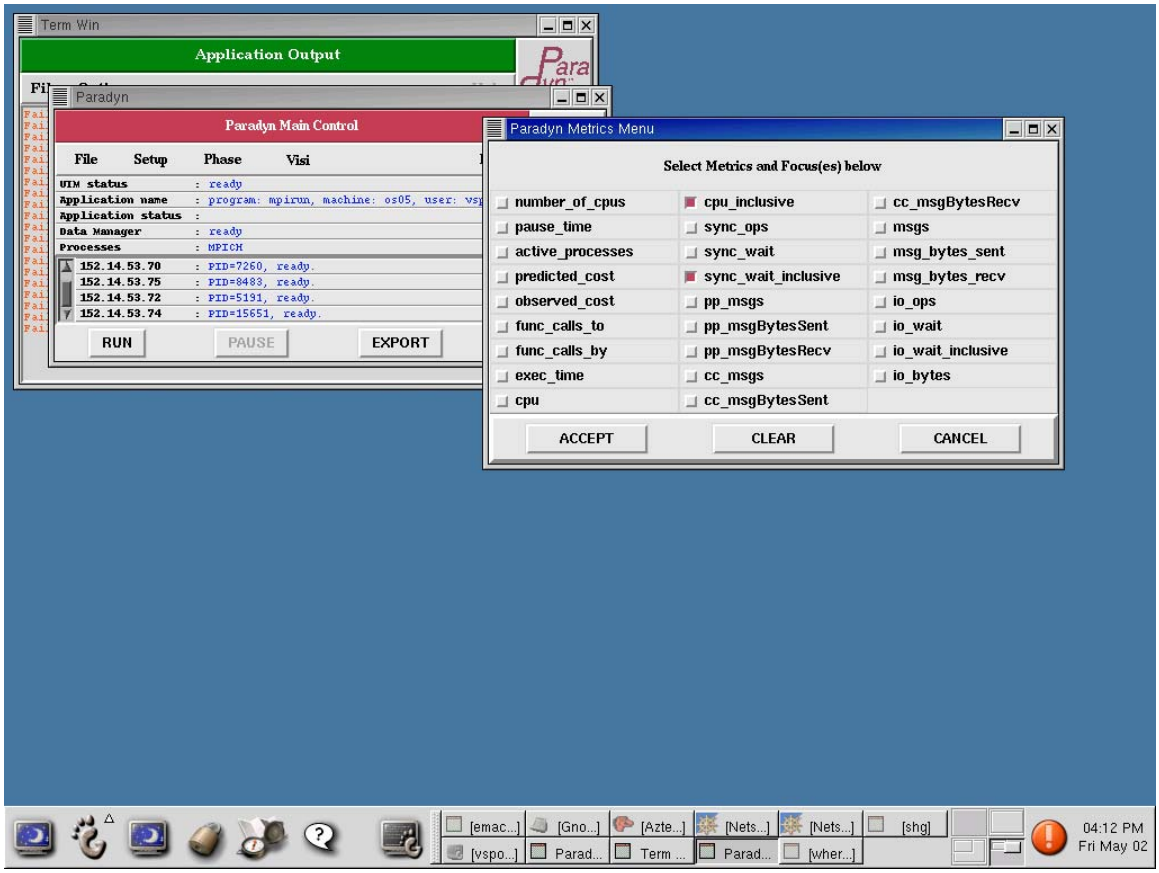


Fig 5: Paradyn Metric options – To plot histogram

The above screenshot is one of enabling the metrics cpu_inclusive and sync_wait_inclusive to be collected for the where axes listed in an earlier step.

After this step, we can run the benchmark under paradyn.

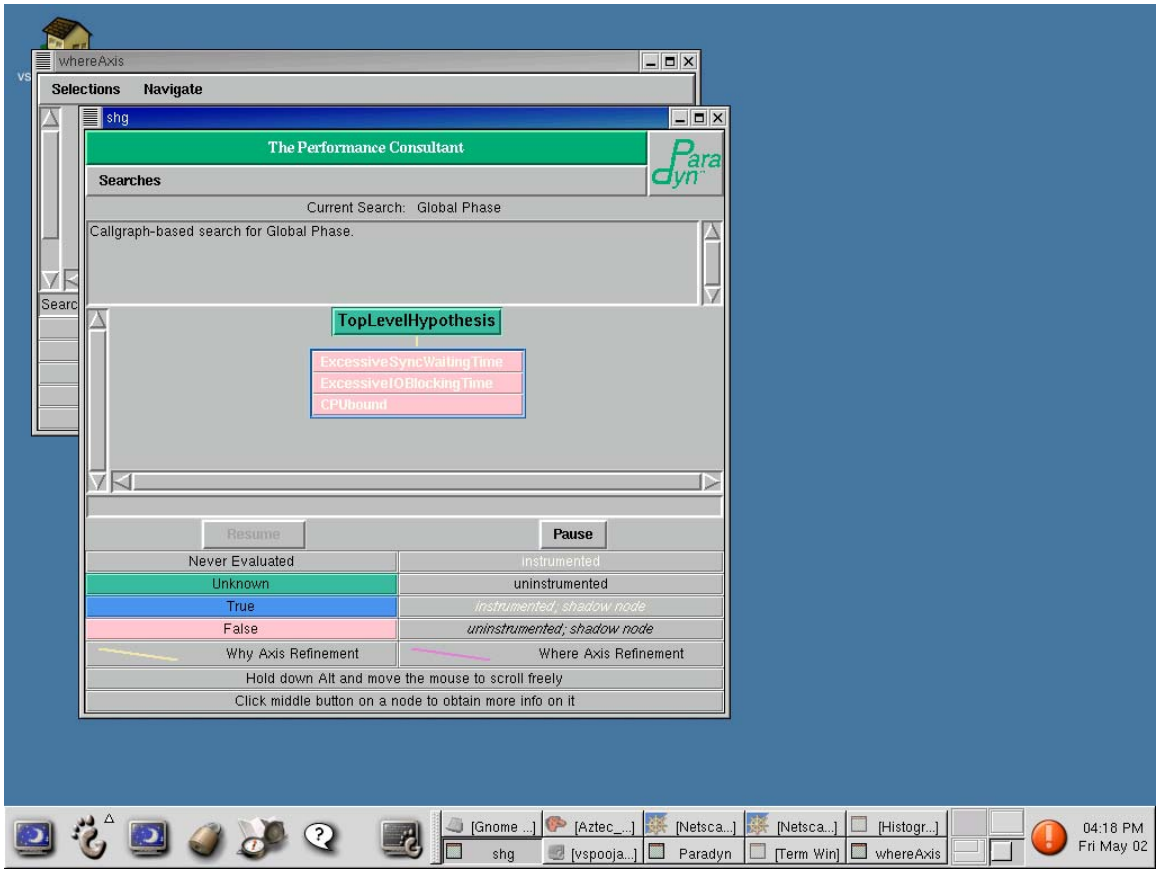


Fig 6: Performance Consulatant – After Program Run

The above screenshot shows the conclusion of the Performance consultant when the application has exited. The above screenshot shows that all of the top-level hypotheses have evaluated to false, implying that the performance consultant is unable to find any bottlenecks for the benchmark.

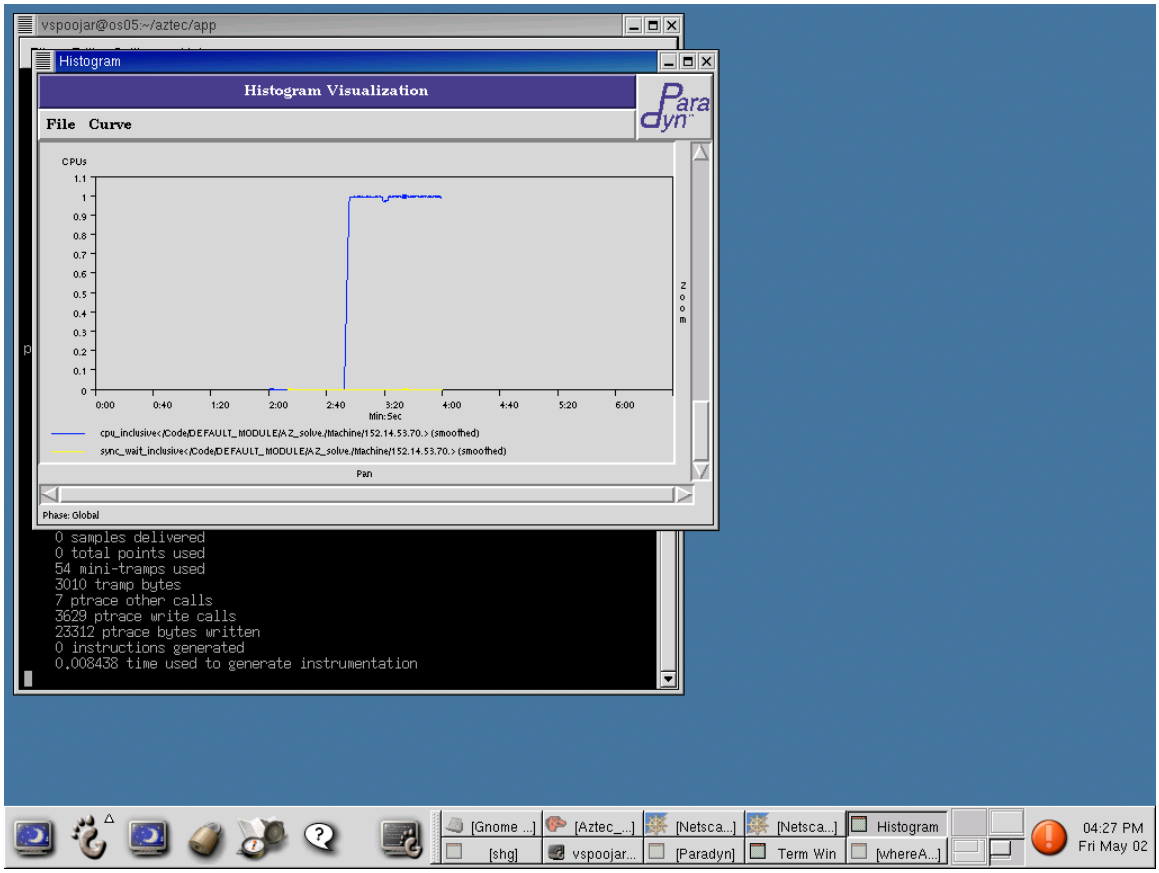
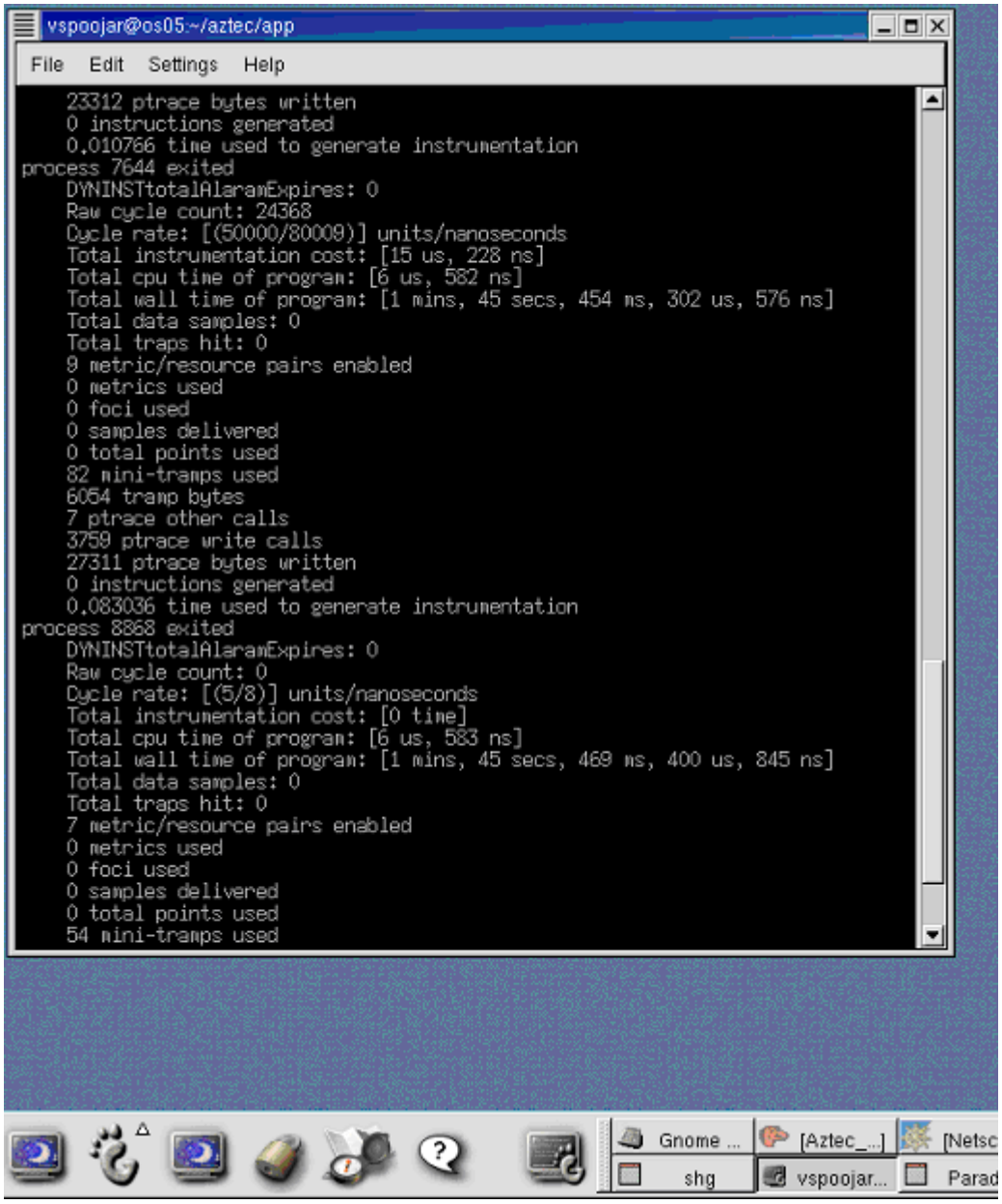


Fig 6: Histogram Plot of CPU time and Sync Wait time

The above screenshot shows the behavior of the benchmark on one of the cluster nodes for the function AZ_solve. This shows that the function is CPU bound as indicated by the blue line and there is no synchronization overhead as indicated by the yellow line.



The above figure shows that Paradyn outputs the overhead due to its instrumentation when the benchmark has finished executing.

So, in conclusion, the performance consultant did not show up any bottlenecks for the benchmark. Consequently, we did not have a starting point to start a detailed manual search by generating Visi graphs for each of the points on the W3 search space. We did however carry out an evaluation for the function AZ_solve by plotting the graphs that is shown in one of the screen shots above.

Suggestions for functionality Improvements to Paradyn:

- Multithreading support is currently only available for Solaris platforms. It needs to be enabled for all the platforms and new abstractions need to be added to the W3 search model to account for multithreading.
- It should be possible to restart applications from Paradyn itself. Currently, it does not allow us to restart the application even the application has finished executing under the tool.

Location of Files in Cluster:

Aztec files: /home/vspoojar/aztec
Test files: /home/vspoojar/aztec/app
Library files: /home/vspoojar/aztec/lib

Paradyn files: /home/vspoojar/paradyn
Binary files: /home/vspoojar/paradyn/bin/i386-unknown-linux2.2
Source files: /home/vspoojar/paradyn/src/core
Test files: /home/vspoojar/paradyn/apps

Readme file for this project: /home/vspoojar/paradyn/README