Applications Working Group

August 19, 1998
Darpa Performance Engineered Systems Workshop
Annapolis

Coordinator: Geoffrey Fox
Participants: Vikram Adve (Rice), Jim Browne (Texas), Frederica Darema (NSF), Apostolos Gerasoulis (Rutgers), Tahsin Kurc (Maryland), John McCalpin (SGI), Graham Nudd (Warwick), Efstathios Papaefstathiou (Warwick), John Rice (Purdue), Subhash Saini (NASA,Ames), Alan Sussman (Maryland), Mary Vernon (Wisconsin) who took excellent notes, and Mary Zosel (DoE,Livermore)

Overall Issues

Here we collect some general remarks on projects
1) All the projects are Total End to End System design (with different emphases) with goals:

- Support hardware system designers
- Support for development of scheduling/system control
- Support for applications developers

2) Some different Design Goals

- Avoid "stupid" application design
- Create near-optimal designs
- Fix existing Programs

3) Don't develop hardware or scheduling in isolation from applications

Benchmarks

There was a long discussion of benchmarks aimed at supporting performance community. First we discussed sharing applications already being investigated by one of the involved groups. This includes:

- ** Sweep3D (POEMS)
- ** MSTAR (POEMS, Warwick)
- ? Titan -- Satellite database (Maryland) I/O Intensive
- ? Virtual microscope (Maryland) I/O Intensive
- Estuary simulation (Maryland and Mary Wheeler at Texas)
- ** Application emulators (Maryland) – easier to distribute
- ** Generic financial (Warwick)
- ** Visualization

Note that symbols above mean

- ** = available
- ? = perhaps unwieldy (large datasets) but application emulator version should be deployable

We briefly discussed other possible sources of suitable applications including:

- ASCI Compact Applications (maybe a year away)
- NASA codes (ARC3D)
- NAS benchmarks

There are some interesting benchmarks sets:

- Specweb
- TPCD

On the following day, we agreed to focus as a group in the near term on two applications (probably Sweep3D and an I/O intensive Maryland emulator) and use them to compare different approaches to Performance Specification Languages discussed below.

Important application characteristics for advancing performance engineering technology: Next we discussed rationale and criteria for choosing an application benchmark suite. We identified the following characteristics.

- Irregular
- I/O (data intensive)
- Adaptive
- not feasible today (A problem that cannot be solved with current technology)
- Important (solves important commercial or technical problem)

In this light, note that SWEEP3D tests memory and computation but not I/O, communication, irregularity or adaptivity. We need new collection of benchmarks with above properties. We identified some generic difficulties:

- Documentation
- Credible complexity versus manageability
- Developer support

This benchmark set is needed to support performance studies of:

- processors, memory, interconnection, I/O, scheduling, runtime support, algorithms

We want benchmarks from variety of programming languages, and different parallelism paradigms

- Fortran, C++, Java…
- MPI,
- DSM, …

As example of unexpected issues that should be covered note that SGI has examples of codes that thrash turning a few percent load imbalance into a larger value as processor that is ahead hammers on memory locations needed by those behind

We suggested following the Perfect Club which introduced a matrix where rows correspond to individual applications and columns to the characteristics.

Matrix:

	Irregular	I/O	Adaptive	Infeasible?	Importance
App1					
App 2					
…					

Could classify from different points of view – the columns could correspond to "features" as above or "algorithmic" structure as in original design of Perfect benchmark set

Who is customer for Performance technology? This allows one to choose mix of applications

- User
- Builder of ad-hoc clusters
- Hardware Vendor SGI IBM Sun ….

Computing Market for SGI and other Computer Vendors and this implies application choice

- 98% commercial (multimedia, web server, email with large images)
- 1% commercial technical
- 0.05% ASCI (larger fraction for SGI)

A lot of commercially important applications are proprietary and so not so easy to put in benchmarks. Web Search is an interesting commercial application.
We discussed fact that industry will not give cost factors and proposed answer:create cost models in which cost factors can be varied

Note recent workshop at Wisconsin on Performance of Web Servers (June 23 98) with
URL for ACM Sigmetrics '98/Performance '98**:** http://www.cs.gmu.edu/conf/sigmetrics98/
URL for Internet Server Performance Workshop:
http://www.cs.wisc.edu/~cao/WISP98.html
URL for Fox's keynote talk: http://www.npac.syr.edu/users/gcf/wisconsinjune98/

Another criteria for choosing applications are the questions users are asking. Mary Zosel gave some examples:

- Thread performance
- I/O performance (e.g. 9hrs of I/O for 4 hrs of computing)
- Java/C++
- Unexpectedly long compute time

There appear to be no tools available to understand poor performance in such areas. This prompted a lively discussion of

- Need for tools available to design it correctly in the first place!
- Versus tools to identify performance problems in existing codes
- Tools aimed at near optimal design versus tools with more modest goal of avoiding "stupid" application design (called identifying bottlenecks by Reed in Delphi talk

Consensus: need new set of benchmarks supporting end to end performance engineering and that have very dynamic behavior and stress other features that stress performance (tools).

Validation and Sensitivity Analysis methodology:

We had a short but lively discussion, which is not done justice by our bland list of issues:

- Compare performance projections against existing systems, get good error estimates
- Component by component approach doesn't work; need to get errors for composites
- Sensitivity analysis is important
- All aspects of this are hard

Performance Specification Languages

There appears to be a convergence to a common representation with all the groups using some form of task graph applied at all levels of design. However it is not clear if people aren't using same words for different things – task graphs at coarse and fine grain are very different. One needs different functionality for nodes in the task graphs for different levels of granularity (1 byte to 64 Kbytes – latter is Maryland granularity, former is Rice). For instance, need address stream to study memory accesses & false sharing, but you don't need this for studying coarse grain I/O performance.

Need convergence on "standard" representation for task graphs in terms of structure and properties. This is essentially what a PSL or Performance Specification language does. Several of the groups have defined a PSL. This must express the task graph or whatever one expresses application and their performance characteristics with. See **http://www.npac.syr.edu/users/gcf/petasimaug98-2** for a desription of PetaSIM PSL.

A PSL must certainly meet some general requirements including.

- Need to express shared memory issues well
- Need to consider both implicit (as in cache use) and explicit data movement – SGI views as very important

Three kinds of customers:

- application designers that have fixed system and so ease of expression of application is most important
- application designers that have system flexibility who need to express application and system flexibly (MSTAR is like this)
- hardware designers who would design against a relatively fixed application set and so issues in ease of expression of application are less important but must express system easily

PSL plays similar role to "design framework" for software development. PSL plays a similar role to SUIF and hardware design languages like VHDL

Applications developers will only use performance specification language if there is a demonstrable payoff in the analyses that it enables and if there is a standard so can leverage investment by users and application designers. On the other hand, there is a tremendous need to rapidly prototype a proposed algorithm e.g., MSTAR

PSL based Tools:

Automatic conversion of Application code → performance specification
Vendor tools to support PSL, analyses

Four Step PSL Adoption Process:

Use current contracts to experiment with different PSL's and do initial comparison
Agree on a PSL
This community produces feasibility demonstrations -- **demonstrations of benefit**
Vendor tools appear supporting API (SGI Considers a standard PSL very important)

Related Relevant activities

- Bart Miller et al.'s efforts (API for distributed instrumentation)
- ARM -- application resource monitoring (Tivoli, HP …) (vendors will report standard information about application behavior) Effort is limited because vendors need to agree on what will be reported
  - more comprehensive set of measures needed
- JINI – Java resource registration and discovery

Education in applied performance modeling/project results is needed. We must Evangelize at Sigmetrics, CMG, Supercomputing conferences, other application areas