Using the Parallel Virtual Machine for Everyday Analysis

Michael S. Noble, John C. Houck, John E. Davis, Andrew Young, Michael Nowak

Kavli Institute for Astrophysics and Space Research
Massachusetts Institute of Technology

ADASS XV San Lorenzo de El Escorial, Spain October 4, 2005
**The Problem**

<table>
<thead>
<tr>
<th>ADS Full-Text Search</th>
<th>#Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel pvm</td>
<td>38</td>
</tr>
<tr>
<td>message passing mpi</td>
<td>21</td>
</tr>
<tr>
<td>xspec</td>
<td>832</td>
</tr>
<tr>
<td>xspec AND (#1 or #3)</td>
<td>0</td>
</tr>
</tbody>
</table>

- Astronomers do employ parallel computing ...
- But not for “standard” fitting/modeling/analysis
- Especially not for interactive exploration
Our Solution

ISIS
S-Lang
PVM
PVM module
+
SLIRP

Scriptable, Modular, Parallel Analysis
The Pieces: ISIS & S-Lang

- All XSPEC models (and more) ... with very fast array math and programmability of IDL / MATLAB
- User models compiled OR scripts (yes, even PYTHON)
- Mature (available > 4 years), reliable, numerically robust
- Designed to be extremely modular and customizable

Almost everything is a module
Most are available outside of ISIS
The Pieces: PVM & SLIRP

- PVM: mature, dynamic, fault-tolerant COW parallelism
- PVM wrapper module just one of scores available
- SLIRP code generator: wrap C/C++/Fortran models

```bash
svoboda% slirp -make kerr.f
Starter make file generated to 'kerr.mf'

svoboda% make -f kerr.mf

svoboda% isis

isis> import("kerr")
isis> kerr
Usage: kerr(float_ptr, long_ptr, float_ptr, ...)
```

ASIDE: Java was tried, but didn’t scale
Case Study I: Kerr Disk Line

- General Relativistic Model
- Previous Implementations Used Look-up Tables (For Speed)
  - Limits Parameter Space
- “On the Fly” Calculation is More Flexible, but Expensive
  - 3 - 4 hours for fit to converge

Sum of Individual Computationally Expensive Pieces

Fortran Model: L. Breneman (UMD)  SLIRP-ed & Parallelized: A. Young & M. Noble (MIT)
Case Study I: Methods

Master / Slave, with light message-passing

```c
public define pkerr_fit (lo, hi, par)
{
    variable klo, khi;
    (klo, khi) = _A(lo, hi); % angstroms to KeV
    return par[0]* reverse(master (klo, khi, par));
}
```

ISIS is not aware that model has been parallelized
Parallelism hidden/achieved within model callback function
Serial/parallel user models installed/invoked same way

```c
add_slang_function ("pkerr", ["norm", "lineE", ...]);
```
Case Study I: Performance

1 Kerr Model Evaluation
Small, Faked dataset

Linear Speedup
Peak speedup = 6.1

Realized Speedup = 4.15

3-4 hour fits reducible to approx 45 mins

14 CPUS: 8.16 s

Interactivity becomes feasible!

1 CPU: 33.86 sec (2GHz Opteron, our fastest)

kerr + power law
12 parameters

Michael S. Noble  ADASS XV  San Lorenzo de El Escorial, Spain  October 4, 2005
### Aside on Amdahl’s Law

<table>
<thead>
<tr>
<th>Description</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runtime on 1 cpu</strong></td>
<td>( T_1 = S + P )</td>
</tr>
<tr>
<td>(( S = ) sequential, ( P = ) parallelizable)</td>
<td></td>
</tr>
<tr>
<td><strong>Ideal runtime N cpus</strong></td>
<td>( T_N = S + \frac{P}{N} )</td>
</tr>
<tr>
<td><strong>Ideal Speedup (no overhead)</strong></td>
<td>( \frac{S + P}{S + \frac{P}{N}} )</td>
</tr>
<tr>
<td>As ( N \rightarrow \infty )</td>
<td>( \frac{S + P}{S} )</td>
</tr>
</tbody>
</table>

30% Efficiency on 14 CPUs

Misleading Because

4.15 / 6.1 = 68% of peak during “normal business hours” (on heterogeneous CPU collection)
Case Study II: Confidence Contours

- 2D contours can be preferable to 1D levels
- ... feasible for “cheap” models
- With parallel evaluation ...
  - ... feasible for even “expensive” models
- Without requiring model parallelization

Supernova SN1006: Joint X-ray, radio, $\gamma$-ray fits
Case Study II: Confidence Contours

Model includes synchrotron radiation component

\[
\frac{dW}{dvdt} = \frac{\sqrt{3}e^3B}{4\pi mc^2} \int dpN(p) \int d\phi \int d\alpha \sin^2 \alpha \ F\left(\frac{v}{v_0\gamma^2\sin \alpha}\right)
\]

Expensive, even with some pre-fabricated tables
Must be computed once per spectral bin
Hundreds per model evaluation, Thousands per fit
Potentially millions per confidence grid

170 x 150 grid: max electron energy X magnetic field strength

Computed in 10 days (20 - 30 CPUs)
6-10 months on single machine

G. Allen et al
2004
Case Study III: ICM Heating in Perseus

Map produced from 10,000 spectral extractions & fits

20+ CPUs

Several Hours (wallclock)

M. Wise et al, 2005 (COSPAR)
Case Study IV: Error Bars

- Independently evaluate each model param
- No excuse for computing this serially!
- Again: models need not be parallelized

Superlinear speedups possible

When i-th slave finds lower $\chi^2$ ...

... terminates other N-1 slaves early

... all restart with new param values
Case Study V: Sequential Models

Shouldn’t it be possible to concurrently evaluate ...

... independent (e.g. non-tied) components of a sequential model?

... entire sequential model over a partition of the bins?

... entire sequential model over individual datasets?
Conclusion: Are we there yet?

Close: Parallelism exploited & most results published ... But some savvy & initiative still needed

- Auto-detect PVM (or maybe MPI) at ISIS install time
- Transparently distribute errors/confidence (NHOSTS > 1)
- ... or independent components of even sequential models
- Script templates OR suite of pre-parallelized models?
- Threaded to use those extra CPUs on newer desktops?

Complex models becoming more common, not less ...

Relevance proportional to size of modern data sets & prevalence of multi-observation/wavelength, VO analysis