

GGF UPDT Program Development Tools Survey

Focus on Grid

Susanne M. Balle
Hewlett Packard
GGF UPDT RG co-chair
Version 0.2



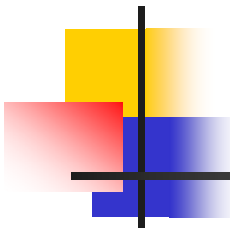
Our goal

- Understand how users develop programs
 - For large scale systems
 - For the Grid
 - Other
- Get more information about the tools they use and if they are adequate
- Understand the challenges encountered when developing apps for large scale systems and the Grid



Where did we distribute the survey?

- Mailing list
 - GGF UPDT RG
 - Globus
 - MPICH G2
 - GGF Apps WG
 - GGF APM RG
 - ...
- Pittsburgh Supercomputing Center
- Colleagues
- Etc.



High level description of the survey

1. User profile: type of apps, research area, etc.
2. Platforms: system architectures, etc.
3. Design cycle: tools used, etc.
4. Program development cycle:
 - languages, parallel programming paradigms, etc.
5. Debugging cycle: tools used, etc.
6. Performance Tuning cycle: tools used, etc.
7. Maintenance and administration: tools used, etc.
8. Other



Summary of results

- Survey responses gathered from February 2003 until August 2003.
- 20 respondents completed the GGF survey
 - 10 completed the “test” survey
 - 10 completed the on-line final survey



Platforms (1/3)

- Writing apps for the 1000+ processors?
 - 45% Yes | 30% No
- Development versus production system
 - 62.5% uses the same system for development and production
 - Varies with the project



Platforms (2/3)

- Users use any machine they can get access to
 - HP AlphaServer SC (Lemieux @ PSC) (>3)
 - Production run 512, 100-600, ...
 - HP Itanium (1), IBM SP (2), Fujitsu VX (1), SGI Origin (2), Itanium Cluster (2), Power4 Cluster (1), IBM p690 (1), 32-processor Linux Cluster (3), 256-processor Pentium cluster (1), 384-processor Windows 2000 cluster (1)



Platforms: conclusions (3/3)

- Users are using all the systems they have to their disposal.
 - No customization of code for specific architecture
 - Applications have to be portable
- “heterogeneousness” is an important factor to consider when developing Grid-enabling apps, middleware softwares and tools
 - SPACI (Southern Partnership for Advanced Computational Infrastructure)
 - 3 HP Alphaserver SC (16 procs), 2 Origin 2000, IBM SP2 (16 procs), IBM SP3 (8 procs), Meiko CS2 (128 procs), 6 Beowulf clusters



Design cycle: conclusions

- Apps design/implementation:
 - Starting from scratch (6)
 - Modifying existing code (6)
- Popular design cycle tools: MATLAB, UML, HTML, and taking advantage of a number of already existing building blocks.
- Rank specific factors to consider when designing applications:
 - Scalability was rated most important
 - Robustness, Fault tolerance, Recovery mechanism, and Modularity seemed important but didn't get priorities assigned to them consistently



Designing apps for the Grid

- Heterogeneoususness: Most important factor to take into consideration
- Compatibility and Flexibility: next highest
- Other mentioned factors: Software Licensing, Interoperability, Resource Management.



Software development environment: conclusions

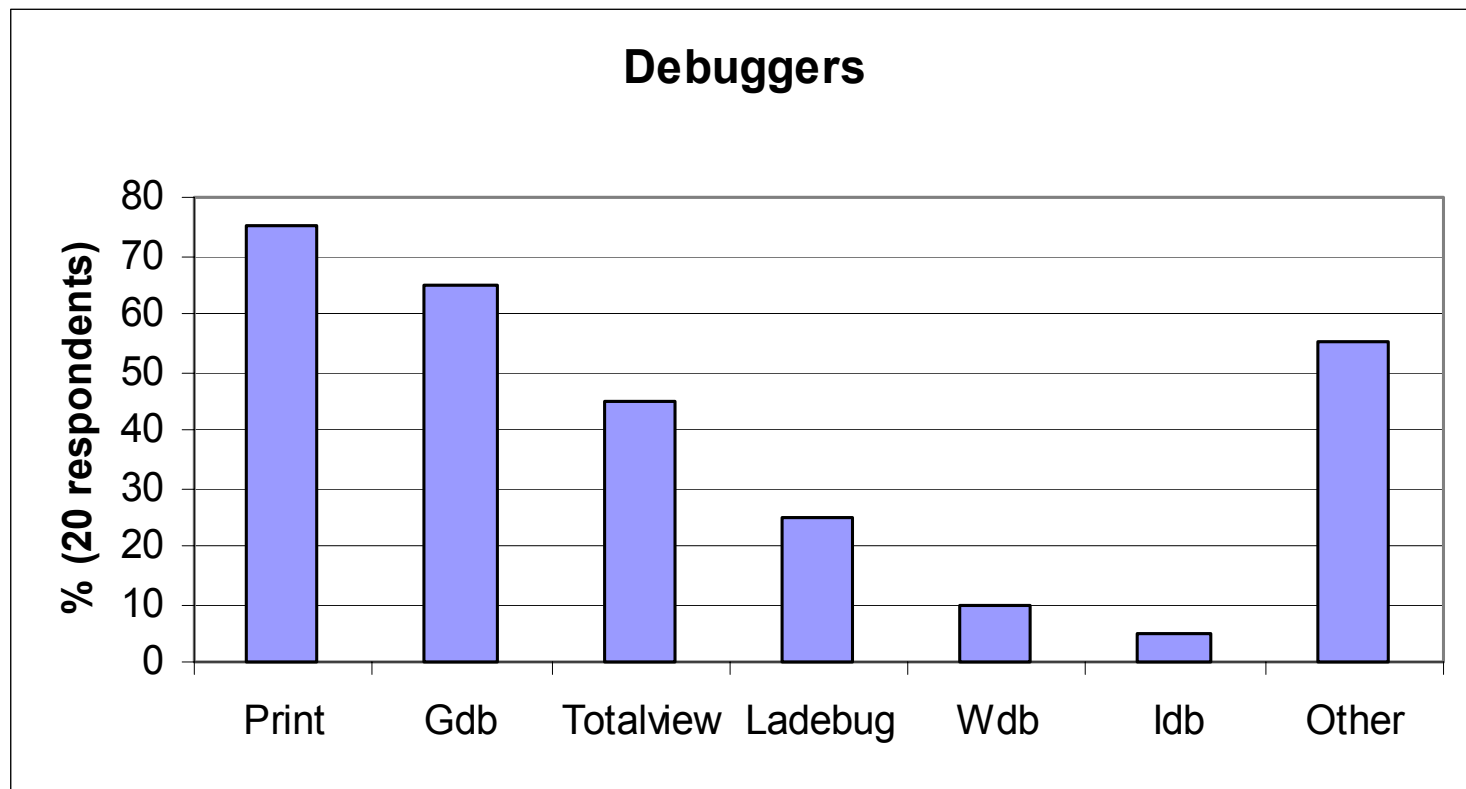
- Respondents are using the common programming paradigms (see Grid Primer, Lee & al) to program for the Grid
- 35% (/20) are looking into Grid enabling their apps
 - Already start 15% | within 6 months 10%
 - Within 12 months 10% | other timeframe 10%
- 20% (/20) uses MPICH G2
- 25% (/20) uses Globus



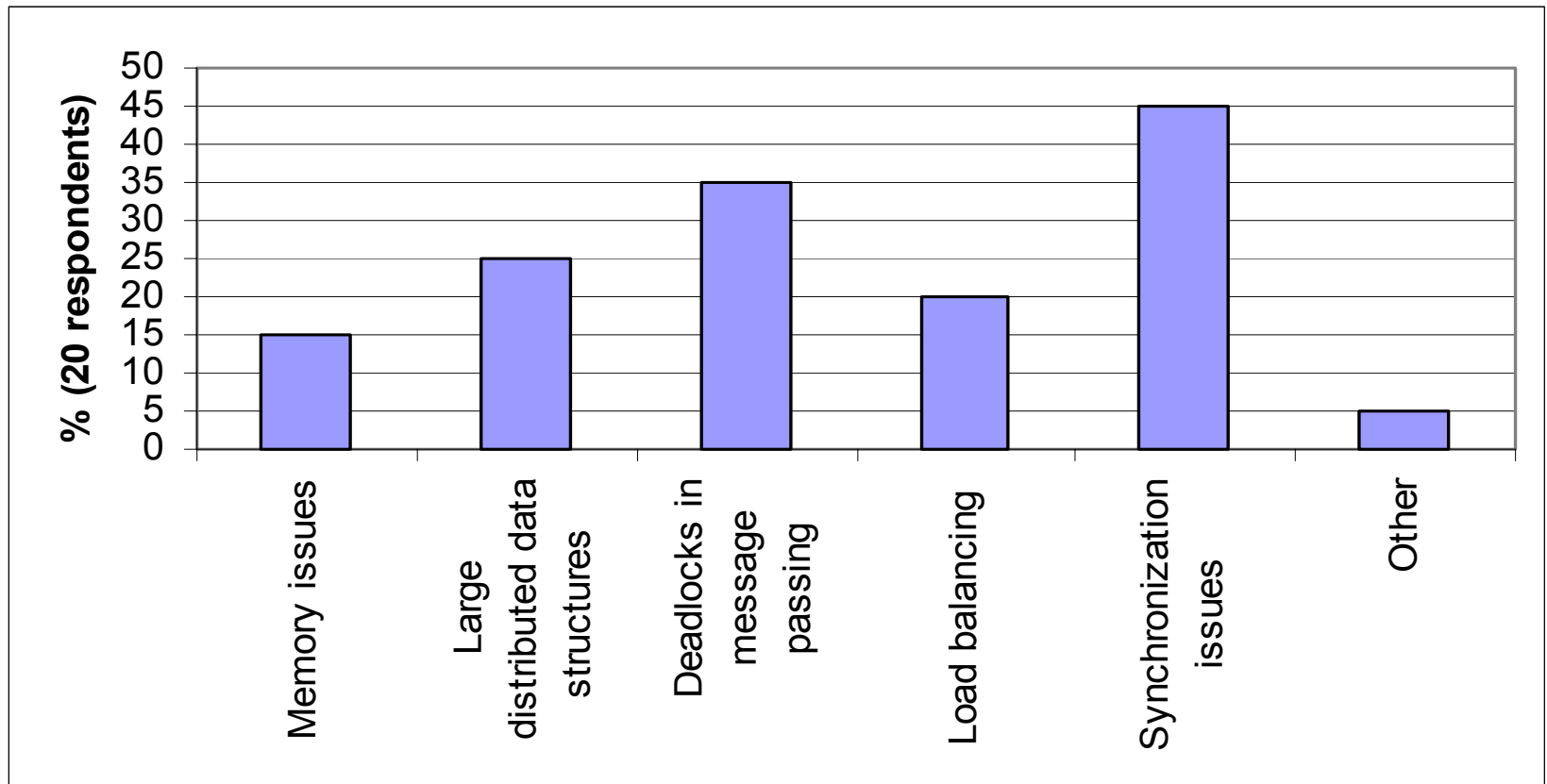
How close are the respondents to developing for the Grid?

- 35% will develop Grid enabled apps within a year
- Grid is still new and not widely adopted
- Not able to gather data that would explain why so few people are running or developing programs for the Grid.
- Our guess: Developing apps for the Grid is too hard and too time consuming. Very few robust and usable Grid-aware tools currently available.

Debugging cycle (1/4)



Debugging cycle (2/4)





Debugging cycle (3/4)

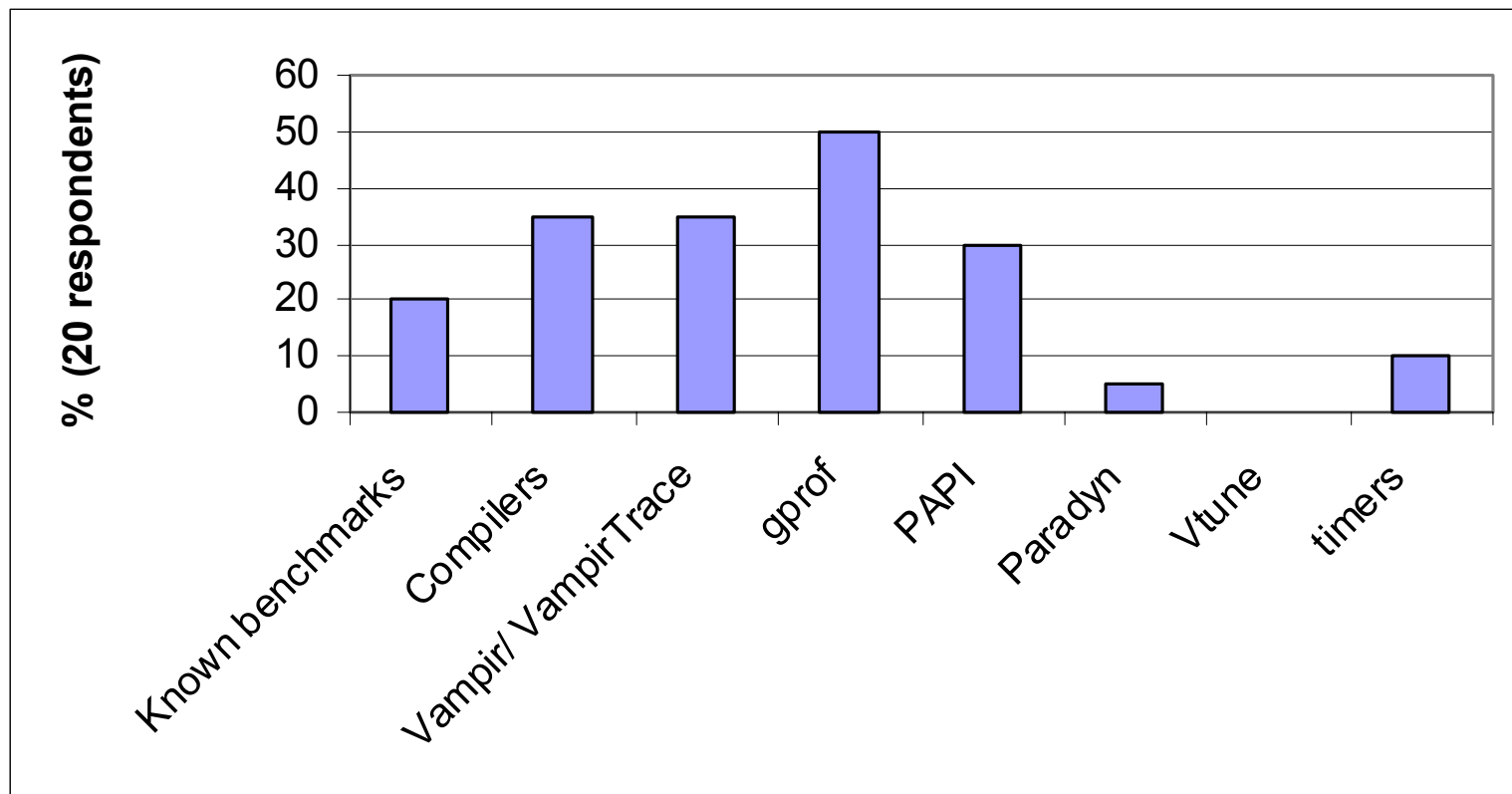
- Respondents use or try to use standard debugging techniques on Large Scale Systems and the Grid.
- Adequate techniques for non Grid-Env.
- Circumvent not having Grid-aware debuggers by login in and attaching to the running to local processes. Not always possible esp. as the Grid becomes more widely adopted.



Debug cycle and the Grid (4/4)

- Additional problems created by the Grid
 - Remote debugging
 - Arithmetic
 - Irreproducibility
 - Dynamic issues
- Debugging by attaching to the local processes
- Debug the same way as we debug multi-process programs

Performance tuning cycle (1/4)





Performance tuning cycle (2/4)

- Example of tuning session (large scale system): (adequate)
 - Performance specifications/benchmarks to set expectation
 - Compilers with code annotations to see how the compiler analyzes the code
 - Pixie to get operation counts
 - DCPI for low-impact profiling to discover where the time is spent
 - Atom tools for post-compile instrumentation to time the code



Performance tuning cycle (3/4)

- **Wishlist:**

- Processor simulator which would show where time is spend (1),
- Automatic performance tuning a la "ATLAS" (1),
- Automatic performance monitoring a la "PAPI" (1), gprof + PAPI hooks (1)



Performance tuning cycle and the Grid (4/4)

- Additional problems created by the Grid
 - Variable latency
 - Bandwidth
 - CPU speed
 - Everything!
- Tune each machine separately
- Grid-aware tools: Marmott, Dimenas



Maintenance and administration: Conclusion

- Tools: (/20)
 - Unix scripts (1)
 - CVS (4) adequate
 - RCS (2) adequate
 - PBS (1)
- In many cases the person working on the code keeps the files



Tools for the Grid: conclusions

- Important that we have adequate tools to allow users to migrate to Large Scale Systems and the Grid.
 - Need to reduce the time it take to develop apps on large systems.
 - Currently too difficult and too time consuming for the average developer.
- Need for Grid-enabled middleware and tools to help users develop their apps for the Grid.
- Globus, GridLab, Cactus, PACX-MPI, MPICH-G2, etc. lead the way by providing middleware that allow user to take advantage of the Grid at a lower cost in time and resources.



Conclusions

- From one of the respondents: “It is indeed curious that scientists who presumably have been drilled in the methodology of rigorous”scientific inquiry for their particular discipline [...] are often very casual when reporting on the performance of the computers they employ in their research.” R. Hockney, “The art of Computer Benchmarking”
- Report available at UPDT website
- www.kvasar.com/GGF/GGF_UPDT.html



Thanks to:

- Sergiu Sanielevici (Pittsburgh Supercomputing Center)
- Jie Song (Nanyang Technology University (NTU))
- Pierre Lagier (Fujitsu)
- Users who completed the survey