Basic Requirements for Systems Software Research & Development

Chris Kuszmaul[1] and Bill Nitzberg[1]

Report NAS-96-014 September 1996

MRJ, Inc.
Numerical Aerospace Simulation
NASA Ames Research Center, M/S 258-6
Moffett Field, CA 94035-1000

parallel@nas.nasa.gov

Abstract

Our success over the past ten years evaluating and developing advanced computing technologies has been due to a simple R & D model. Our model has three phases: (a) evaluating the state-of-the-art, (b) identifying problems and creating innovations, and (c) developing solutions, improving the state-of-the-art.

This cycle has four basic requirements: a large production testbed with real users, a diverse collection of state-of-the-art hardware, facilities for evaluation of emerging technologies and development of innovations, and control over system management on these testbeds. Future research will be irrelevant and future products will not work if any of these requirements is eliminated.

In order to retain our effectiveness, NAS must replace out-of-date production testbeds in as timely a fashion as possible, and cannot afford to ignore innovative designs such as new distributed shared memory machines, clustered commodity-based computers, and multi-threaded architectures.

1. Work performed under NASA contract NAS2-14303

1.0 Introduction

The parallel systems (PS) group was created in 1989 with the charter to ensure that emerging computing technology (expected in five to ten years) will efficiently support a diverse scientific workload.

To this end, we develop system software innovations that enable use of parallel computers. Our innovations have come in three areas:

- Operating systems (reliability, administration, single system image)
- Resource management (batch, scheduling, accounting, limits, checkpointing)
- Parallel I/O (file and network)

In each area, our most lasting contributions have been standard interfaces, and new algorithms. Interfaces and algorithms transcend a single computing platform or programming paradigm, and promote innovation by solidifying solutions to well-understood problems, allowing researchers to attack new areas.

While the generic concept of a standard, or an algorithm is abstract, a specific standard or algorithm requires a concrete grounding in the real world problems of real users on real systems.

The PS group engages in an ongoing research and development cycle for systems software (see Figure 1).

```
LOOP:
    Evaluate the State-of-the-Art
    Identify problems; create innovations
    Develop solutions improving State-of-the-Art
```

FIGURE 1. Systems software research and development cycle

Combining our long term view with the realities of current systems has allowed us to develop and facilitate standards for message passing (MPI), I/O (MPI-IO), scheduling (psched), as well as design better algorithms in these areas (e.g. collective buffering). Developing these solutions required control over a diverse collection of state-of-the-art hardware for evaluation and development and a production facility for testing and verification.

Our experience demonstrates that each of the following is a requirement for successful research and development.

- large production testbed with real users,
- diverse collection of state-of-the-art hardware,
- facilities for evaluation and development, and
- control over system management.

In section 2.0 we cover the need for a large production testbed capable of sustaining a real user base. In section 3.0, we outline the need for diversity in computer architectures and vendors. In section 4.0, we show the need for evaluation and development facilities. Finally, in section 5.0, we detail the reasons for maintaining control over system management.

2.0 Large Production Testbed

We loosely classify computer systems by performance and workload into three types: evaluation systems, development systems, and production systems.

A production testbed is required to sustain our focus and ensure the quality of the systems software. It uniquely provides the full complexity of a real supercomputing environment. In particular, a production testbed serves dual purposes of:

- evaluating the state-of-the-art to identify problems, and
- verifying that developed solutions and innovations work.

Without a production testbed, systems software requirements cannot be identified, and certainly can never be evaluated for further innovations that would be relevant to real users. Further, to develop systems software that can run on large systems, a large system must be at our disposal to avoid errors of scale in the design and implementation of that software.

Numerous examples exist where the absence of a production testbed has resulted in lost time (A1, A2, A3, A4), and where the failure to have a large system available to reduce potential errors of scale resulted in disaster (A5, A6, A7, A8).

3.0 Diverse Collection of Hardware

To sustain our research, we must have a diversity of computer vendors and architectures at our disposal.

Ten years from now, a mature high performance supercomputer will run scientific applications at a sustained rate near one hundred teraflops. The nature of the computing system that supports this level of performance is unknown at this time. It may reflect the architecture of a cluster of shared memory processors (SMPs) such as the Newton or Davinci clusters that reside presently at NAS. The system may reflect the distributed shared memory architectures represented by the Stanford Flash, or MIT Alewife projects. The system may consist of a cluster of commodity CPUs and network components. The system may be a classical parallel vector supercomputer such as the VonNeumann machine at NAS, or a parallel computer such as is represented by the Babbage machine at NAS.

NAS has succeeded by spanning the above kinds of technologies, avoiding the pitfalls of depending on a single approach (A9), demonstrating the viability of alternative approaches (A10).

4.0 Evaluation and Development Facilities

Before NAS invests in a large scale system, that system must be at least partially proven with an evaluation testbed. An evaluation testbed is a small system with the purpose of evaluating a new technology.

Evaluating systems before making a large-scale purchase has proven effective (A11), while the lack of prior evaluation has shown poor results (A12).

A development testbed allows the group to create and experiment with systems software innovations without disrupting other pathfinding research and development efforts. A development testbed is isolated from the set of applications users to insulate those users from the initial instabilities of experimental systems software.

With development testbeds in place, NAS has made a positive difference in system software (A13). Further, development work performed on a production system can not only drive users away by degrading system performance, but can also result in disastrous consequences (A14).

One alternative to a development testbed that does not drive away users is to not do development. However, the lack of a development environment squelches innovation, and hinders state-of-the-art advances in systems software (A15).

5.0 Control Over System Management

For the evaluation, development, and production testbeds to serve their respective functions, NAS must maintain direct administrative control over these testbeds. This means NAS personnel must have "root" privileges as well as physical access to the hardware. Lack of this kind of direct control can impede every level of research.

Direct control is required for low-level systems software development. For example, the ability to reboot, access the system console, and reconfigure hardware are crucial (A16, A17).

Without control over system management, NAS would be unable to perform basic debugging and troubleshooting tasks, such as isolating hardware and software errors (A18). Further, lack of control greatly increases development time, resulting in delays for both users and developers (A19).

Finally, human factors issues, such as the responsibility which comes with management, are vital. In particular, "pride of ownership" fosters innovation as well as quick problem resolution.

6.0 Conclusion

The PS group at NAS can generate useful research and development results only by retaining:

- a large production testbed with real users,
- a diverse collection of state-of-the-art hardware,
- facilities for evaluation of emerging technologies and development of innovations, and
- control over system management on these testbeds.

Eliminating any of these requirements makes it much less likely that future research will be relevant, or future products will work as designed.

In particular, NAS must replace the out-of-date production testbed (Babbage) in as timely a fashion as possible with a new production testbed. Further, NAS cannot afford to ignore innovative designs such as the new distributed shared memory SGI machines, clustered commodity-based computers, and multi-threaded architectures.

Appendix A: Examples

A1 When initially installed, the SP software from the vendor failed immediately, because it had not been tested on a production system.

A2 PIOFS failed when more than one user attempted to employ it on the SP2 in production mode.

A3 Thinking Machines released a set of numerical library functions that returned zero in all cases—a mistake found only when production users at MRJ attempted to employ those libraries.

A4 The Cray 2 compiler had been tested on only two codes when delivered to NAS—the production environment at NAS motivated an accelerated, focused development effort to correct errors found in the compiler when hundreds of users began to call on it.

A5 In 1989, a major engineering investment by a parallel computer vendor was squandered by trying to directly adapt a piece of software to a supercomputer that ran one thousand times faster than the computer on which the software was originally developed. A one second sleep was required by the software to prevent the system from crashing, negating a factor of one hundred of the potential improvement.

A6 In 1988 and 1989, the CM-2 front end, and the SRM on the iPSC/860 hindered performance of the larger computers because of systems software designed on the slower front end computers.

A7 In 1993, OSF/1 was developed on clusters of 386 PCs, with ethernet connections. When ported to a Paragon computer, NORMA could no longer function because the scale of the memory size, processor speeds, and network performance were dramatically incompatible with the implementation.

A8 When modifications were made that allowed a 66 node Paragon to work, the 208 node version of the machine still failed because of further issues of scale.

A9 NAS avoided the pitfall of depending on CMF—the version of FORTRAN specific to single instruction multiple data (SIMD) computers that are now defunct by maintaining active research in multiple instruction multiple data (MIMD) computers, and the corresponding model for computing known as message passing.

A10 NAS demonstrated the viability of message passing as an alternative to SIMD because, not only was the CM5 at NAS, but so was the iPSC/860. The iPSC/860 computer hardware is now old technology, but the systems software developed on it has allowed NAS to forge further ahead on new systems.

A11 NAS succeeded in this strategy by purchasing a small Cray2, which evaluated well and resulted in the successful purchase of a larger system.

A12 NAS encountered problems when it acquired a 208 node Paragon system without an evaluation testbed as a proving ground. The operating system crashed on an hourly basis. An evaluation system would have exposed some of the aspects of this defect.

A13 In 1984, NAS identified a need for UNIX to run on the Cray-2. By working with Cray to allocate computing resources specifically and exclusively for development, NAS was able to facilitate UNICOS.

A14 150,000 user files were lost when a newly developed, tested (but not tested on a production system) Amdahl striping filesystem was installed on a production platform. The Pancho storage system, installed in 1993, now provides a development environment for large scale file systems, and at this writing is in use for debugging a new filesystem release.

A15 In the absence of a separate parallel development environment, the group has not been able to investigate gang scheduling methods because such an effort would disrupt the SP2 workload.

A16 In order to improve performance of the iPSC/860 parallel file system (CFS), I/O node cables were physically moved to determine the configuration which minimizes contention.

A17 Early integration and testing of SUNMOS on the Intel Paragon required holding a "light pen" up to LEDs on the front panel of the system for debugging. SUNMOS delivered peak network bandwidth and significantly more available memory than the Intel-delivered OSF/1 AD operating system.

A18 On the Davinci testbed, the MPI interface is in a state of flux that depends on how well the HIPPI cables are seated in their sockets. Direct access to the HIPPI cables allowed NAS to quickly isolate a bug in the device driver.

A19 Mpirun took 10 times as long (1 month rather than three days) to port to Newton as compared with porting to other systems where the developers had root access. Installation of mpirun at the Maui High Performance Computing Center remains incomplete—a case where both root access and physical proximity are absent.

	# NAS TECHNICAL REPORT
	Title: Basic Requirements for Systems Software Research & Development
	Author(s): Chris Kuszmaul and Bill Nitzberg
Two reviewers must sign.	Reviewers: "I have carefully and thoroughly reviewed this technical report. I have worked with the author(s) to ensure clarity of presentation and technical accuracy. I take personal responsibility for the quality of this document." Signed: [illegible] Name: Sam Fineberg Signed: [illegible] Name: Ian Stockdale
After approval, assign NAS Report number.	Branch Chief: Approved: [illegible]
Date: 9-18-96	NAS ReportNumber: NAS-96-014