Skills & concepts for research projects in distributed web-based computing
A guide for research assistants

Dr. Daniel Andresen
October 5, 2000
Updated August 7, 2001

Remarks: Most of my research projects tend to involve communicating distributed systems. These tend to be written in some combination of C/C++ (where high speed or low-level control is needed) and Java (for portability and built-in Web support). Below are listed a number of specific skills which are useful when working in the high-performance distributed computing research. However, even more valued are the mental attributes: a willingness to work hard; a mental picture of "why" we are doing something, rather than just "what;" a raging curiosity about everything, but especially a desire to know why the system is behaving the way it does and how it could be improved; a methodical, careful programming style which plans for the future; and, finally, a willingness to experiment, to try things out.

Thoughts & suggestions:

1. "Don't bring me problems, bring me solutions." Have several avenues of attacking a problem before bringing the problem to me. Explore as many of these as seems feasible given constraints of time and resources.
2. Don't be afraid to try things. If we knew what we were doing, it wouldn't be research. It's only a computer! If you break it, we'll restore the system from a backup.
3. Speaking of which, back up early & often. Use a version control system (RCS/CVS) or something similar so you can undo your changes.
4. Form is, unfortunately, almost as important as content. **Be prepared to write clearly & well about your research.** PowerPoint is becoming a necessary skill.
5. Very rarely will you have the tool to magically convert data from your source program to your destination. Learn to chain programs together to get the results you want.
6. In fact, spend lots of time searching for someone else's code to borrow, if possible. Don't recreate the wheel – spend your effort where your new idea is implemented, not on building its infrastructure that everyone else has already completed.
7. Crashes during a demo are downright embarrassing, but cool demos are GREAT for selling your research! So keep around a stable "release" version for demos, while continuing to work on a development version.
8. There is no substitute for hands-on experience.
9. Don't be stopped by lack of resources. If you need books, software, or hardware, let me know.
10. I am neither omniscient nor telepathic. If you don't tell me something, I probably won't know it.
11. Be honest. If you're not making progress, let's work on it together. Don't try to hide it.
12. Your position as a research assistant is a real job, with real responsibilities, not an entitlement. I expect you to work hard for the hours I'm paying for.
13. That being said, strive to maintain a balance between work, school, and play. People make stupid decisions when they are dead tired. Problems often disappear quickly after a good night's sleep.

Skills & knowledge:

1. Writing well
   a. See *On Writing Well*, by William Zinsser. Read this!
   b. Latex + Bibtex can save hours of time in the end.
2. Java programming skills
   a. Look at the CIS200 text
   b. www.javasoft.com has a good tutorial
3. C/C++ programming skills
   a. Take 208/209/520/540
4. Unix familiarity (Solaris & Linux)
   a. Shells & shell programming
   b. Unix IPC/RPC methods
5. Computer architecture – processors, memory & I/O busses (relative speed), memory hierarchy
   a. *Computer Architecture: A Quantitative Approach*, Hennessey & Patterson
6. Networking
   a. sockets/RPC/Java RMI/MPI/PVM
   b. networking hardware – Ethernet(s), TCP/IP, Active Messages
      i. "Active Messages: a Mechanism for Integrated Communication & Computation," by von Eicken, Culler, Goldstein, & Schauser.
7. Parallel programming
   a. Threads (e.g., pthreads) & processes
   b. Synchronization (monitors/semaphores/locks)
   c. Fine- vs. course-grained parallelism; computation/communication ratio
8. High-performance computing – multiprocessing, caching & the memory hierarchy
   a. *Hierarchical Tiling*, by Carter & Ferrante at UCSD
9. Distributed computing architectures – Beowulf clusters, CORBA, message-passing vs. distributed shared memory, "The Grid"
   a. *The Grid*, by Foster & Kesselman
   b. www.beowulf.org
10. XML and its uses
    a. *Professional Java XML Programming*, by Nakhimovsky & Myers
    b. CIS726