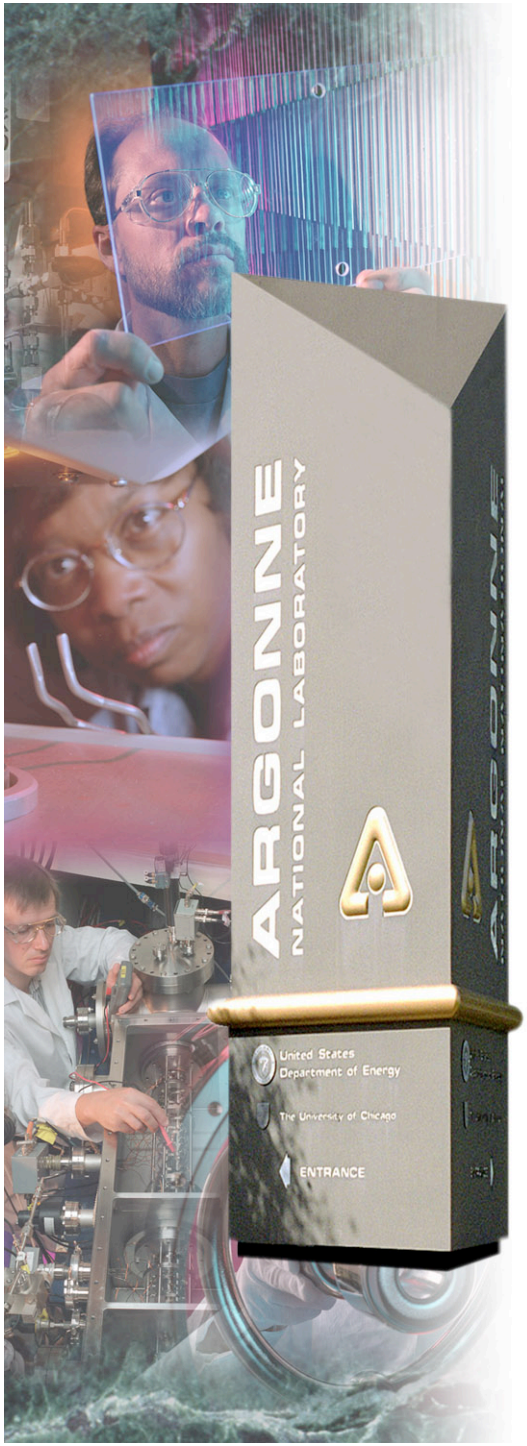

Beware of What You Wish For

William D. Gropp
Mathematics and Computer Science
www.mcs.anl.gov/~gropp

Argonne National Laboratory

Office of Science
U.S. Department of Energy

A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago

What is a file?

Container of data, often indexed by name

- **But also:**
  - Usually implies strong consistency requirements:
    - *When a write to file completes,* any *system anywhere that reads from the file gets the new data (stronger than sequential consistency)*
    - *Extreme consequences for parallel access, difficult to use caches to optimize (and impossible without some cost)*
    - *NFS doesn't implement POSIX semantics for this reason*
      - Why is POSIX required for high-performance file systems when it is considered too expensive for the file system used to edit files and read email?

HPCS Requirements

- **See, for example, http://www.nitrd.gov/subcommittee/hec/workshop/20050816_storage/talks/koester.pdf**
- **How do you determine requirements?**
  - Find out what users need, or
  - Find out what they are doing now, and multiply by one guess as to future system size, or
  - Let users do the multiplication, based on what they think the future system looks like
- **Resulting in these:**
  - 32K file creates/s
  - 1 Trillion files
  - 10K metadata operations/second
  - 30 GB/s (easy!)
    - *But stay tuned…*

32K file creates per second

- **Why are separate directory entries required?**
- **Are files unrelated or are they (or some subset of them) part of a related cluster of data objects?**
  - Alternatives are 1 file (classical, Parallel I/O) or 1 container (new API, semantics tuned for real requirements)
  - Note that the separate file approach introduces many scaling problems, starting with the description of the data set by enumeration
- **Is this really a database operation?**
  - Possibly with different types of entries from classic database

1 Trillion total files

- **Why are these separate files?**
  - Many problems with separate files.
    - *Enforcing POSIX atomicity (including directory updates).*
    - *No efficient search API for directory operations in POSIX.*
    - *Can't use 32-bit int for inodes.*
- **Is this 1 trillion records?**
  - What are the operations on these records?
  - What are the atomicity requirements?
  - A (custom) database system built from low-level I/O (like real databases) might be a more effective and realistic solution
    - *What are the requirements for* **that***?*

1 PF => 1PB main => 1 TB/s for 1000 seconds for checkpoint

- **A checkpoint requirement can lead to massive bandwidth requirements**
  - 1PB of memory may cost way more than you expect; memory prices may be leveling off
  - There is a danger that the requirement becomes decoupled from the actual system (e.g., change 0.1 PB memory for cost, but retain 1TB/s for checkpoint)
- **Is this continuous or burst?**
  - Continuous – require enough bandwidth to file system hardware (e.g., spinning disks)
    - *1 TB/s / (20 MB/s/disk) = $10^{12}/2\times10^{7}$ = 50000 disks*
      - Use *sustained transfer rate,* not *burst transfer rate*
    - *Buy stock in Segway or stock up on roller blades*
  - Burst – option to buffer in higher speed storage (faster than disk semiconductor memory)
    - *But still need order of 1PB memory of some kind*

- **Is the checkpoint for all 1PB of memory?**
  - May not need to dump entire memory to disk, at every checkpoint
    - *Can reduce requirements by an order of magnitude*

Summary

- **Don't say "file" when you mean "data"**
- **Be careful what you wish/ask for**
  - Who is going to answer "Do you want a POSIX file system" with "no"?
  - But what if you add in the consequences of that choice?
- **POSIX consistency is a very strong requirement**
  - Hard to implement both correctly and fast
    - *NFS does neither*
- **Simple scaling of current approaches to the next generation of machines leads to unnecessarily expensive requirements**
- **We shouldn't be talking about files at all**
  - Rather, persistent data with defined consistency rules and defined persistency (checkpoint data may need less persistence than science data; other data may be immutable)
  - RENCI Petascale Data System is an example of this approach