HPC storage benchmarking

Mike Mesnier (Intel/CMU)

James Hendricks, Raja R. Sambasivan, Brock Taylor (Intel),
Matthew Wachs, Greg Ganger, Garth Gibson

Parallel Data Lab
Carnegie Mellon University

Motivation

- HPC apps must be smart about their I/O
  - Massively parallel access
  - Collective I/O, strided accesses
  - May adapt to strengths of the storage system
- Consequently, storage system evaluation
  - Can be difficult for complex applications
  - Can be expensive (time and money)
- HPC storage benchmarking is one solution

Generating *representative* I/O is the challenge

Representative I/O??

- Traces
  - Asynchronous (deterministic) playback
  - Increasing playback speed is not realistic
- Micro-benchmarks
  - Good for testing specific scenarios (e.g., iozone)
- Macro-benchmarks
  - Useful, but too domain specific (e.g., TPC-C)
- Is any *one* benchmark "representative?"
  - Computational chemistry, biology, earth sciences , oil/gas, pharmaceuticals, … (probably not)

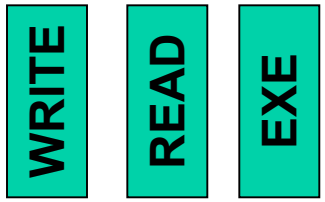
Our approach: "rapid prototyping"

1. Profile the primary I/O phases of an app
   - Parallelism, write ratio, randomness, etc.
2. Automatically generate I/O processes
   - A distributed workload generator (e.g., b_eff_io)
3. Generate I/O against system
   - Good for measuring first-order effects

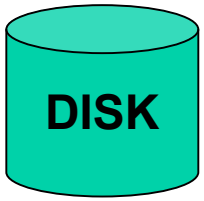
RP is common among distributed systems:

- Graphical tools for visualizing/analyzing workflow
- Languages for rapid prototyping (e.g, EMSL)
- Compilers to generate synthetic processes

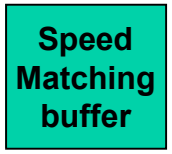
Example icons for rapid prototyping



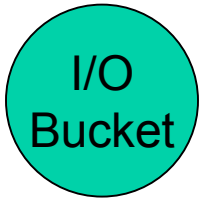
- Read, execute or write process



- Input or output



- Allows for parallelism between two processes

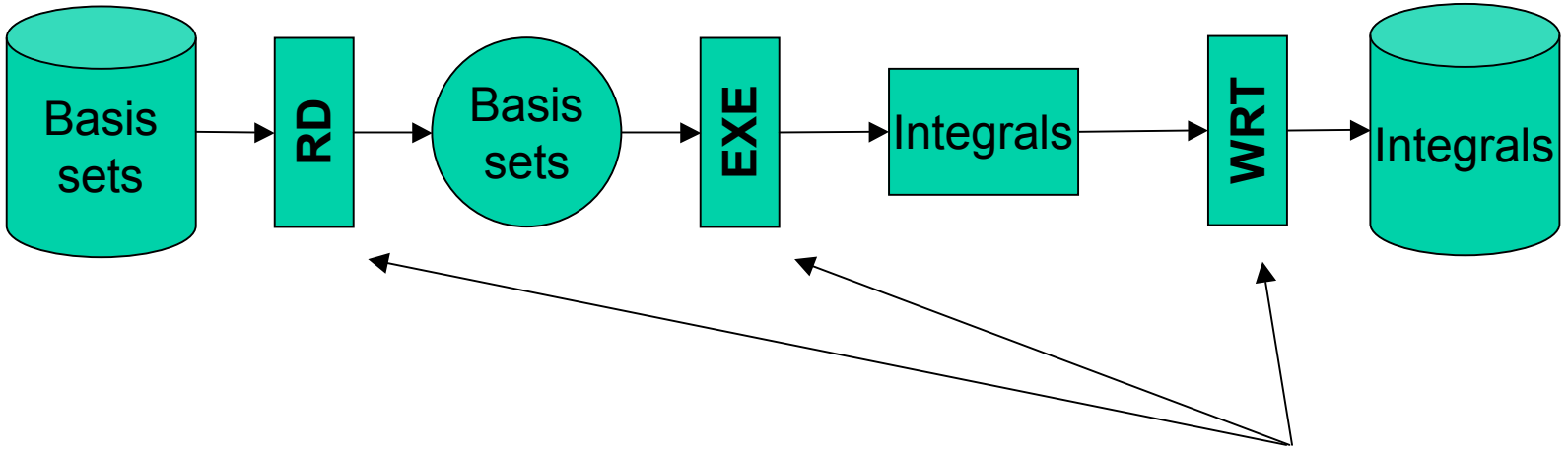


- Producer/consumer buffer (no parallelism)

- IPC between two nodes

Example (computational chemistry)

- For all nodes do
  - Read in basis sets (atomic orbitals)
  - Compute atomic integrals
  - Write atomic integrals to disk



Must specify characteristics of each process (e.g., request size, access pattern, passes over data)

Next steps

- Select a modeling environment
  - Graphical tools, language, compiler
  - E.g., FileBench from Wednesday's BOF
- Extend modeling environment for HPC
  - Multiple processes, parallel I/O, barriers and synchronization, strided access, …
- Provide "reference" profiles for common apps
  - Computational chemistry, oil/gas, etc.

Questions?