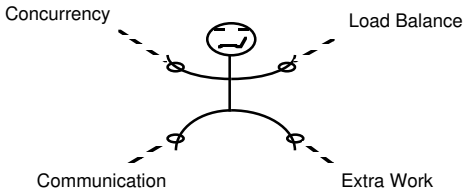
Towards Workload-Driven Architectural Evaluation

- Challenges
- Workloads
- Scaling

© Culler 1995

Recap: Fundamental Tension in Prog. for Perf.



- Programming / compilation must balance factors
- Architecture can lesson the “gradient”
  - **fast communication (low latency, high bandwidth)**
    - » **allows finer grain, smaller transfers**
  - **convenient abstractions (shared data structures)**
  - **automatic data migration and replication**
- Example 2: n-body problem (see slides from lecture 5)

© Culler 1995

The Challenge of Workload-driven Evaluation

- Difficult even for uniprocessors
- Much more difficult for multiprocessors
  - **Representative workload?**
    - » **many specialized areas**
    - » **rapid change**
  - **Software model has not stabilized**
  - **Many architectural and application degrees of freedom**
    - » **huge design space**
  - **Impact of architectural and program parameters can be huge**
  - **Little consensus on appropriate metrics**
  - **Simulation is expensive**
    - » **slow-down applified by P**
    - » **huge problem sizes**
    - » **realistic configurations and sensitivity analysis difficult**

© Culler 1995

Bailey's 12 ways to fool the masses

- Compare 32-bit results to others' 64-bit results
- Present inner kernel results instead of whole application
- Use assembly coding and compare with others' Fortran or C codes
- Scale problem size with number of processors, but don't tell
- Quote performance results linearly projected to a full system
- Compare with scalar, unoptimized, uniprocessor results on Crays
- Compare with old code on obsolete system
- Use parallel implementation as base instead of best sequential one
- Quote performance in terms of utilization, speedups/peak MFlops/$
- Use inefficient algorithms to get high MFlop rates
- Measure parallel times on dedicated sys, but uniproc on busy sys
- Show pretty pictures and videos, but don't talk about performance

© Culler 1995

Issues in Workload-Driven Evaluation

- Choosing workloads
- Scaling workload and architecture
- Picking parameters and analyzing sensitivity
- Metrics, and presenting results

© Culler 1995

Choosing Workloads

- Representativeness
- Adequate concurrency
- “Batch” versus time-sharing
- Do they stress the features of interest?
- Applications v. kernels v. toys v. microbenchmarks

© Culler 1995

Representativeness

- Should adequately represent domains of interest:
- **Scientific:** Physics, Chemistry, Biology, Weather ...
- **Engineering:** CAD, Circuit Analysis ...
- **Graphics:** Rendering, radiosity ...
- **Information management:** Databases, transaction processing, decision support ...
- Optimization
- **Artificial Intelligence:** Robotics, expert systems ...
- **Multiprogrammed general-purpose workloads**
- The operating system

© Culler 1995

Concurrency

- Should have enough to utilize the processors
- Should maintain adequate load balance
  - **need to know what programs and configurations do not**
  - **may dominate impact of communication performance, architectural efficiency, etc.**
- At least, should isolate performance limitations due to program characteristics that a machine cannot do much about (concurrency) from those that it can.

© Culler 1995

Batch versus Time-shared

- Batch (one application at a time):
  - **+ easier to construct workload**
  - **+ cleaner, controlled environent**
  - **+ easier to interpret results**
  - **+ repeatability**
  - **+ can represent space-shared multiprogramming**
  - **-- *many multiprocessors are time-shared***
- **Time-shared:** opposite of above:
  - **+ complex and realistic**
  - **-- difficult to construct "representative" workloads**
  - **-- difficult to interpret results**

© Culler 1995

Does it Stress the Right Features?

- *compute v. memory v. communication v. I/O bound*
- *local memory bandwidth*
- *communication bandwidth*
- *communication latency*
- *fine-grained or coarse-grained*
- *synchronization*
- performance under contention
- communication patterns
- Choose workloads that cover a range of properties

© Culler 1995

Appls / Kernels / Toys / Microbenchmarks

- *Examples*:
- *Complete Applications:* ocean simulation, crew scheduling
- *Kernels:* matrix factorization, depth-first search
- *Toys:* n-queens
- *Microbenchmarks*: message send, remote read

Realistic
Complex
Higher level interactions
Are what really matters

Easier to understand
Controlled
Repeatable

Use kernels and microbenchmarks to gain understanding, but applications to evaluate effectiveness and performance

- don't forget to evaluate the applications too.

© Culler 1995

Workload/Benchmark Suites

- Numerical Aerodynamic Simulation (NAS)
  - **pencil and paper benchmarks**
- SPLASH
  - **shared address space parallel programs**
- ParkBench
  - **message-passing parallel programs**
- ScaLapack
  - **message-passing linear algebra routines**
- TPC
  - **transaction processing**
- HINT
  - **auto-scaling radiosity benchmark**

© Culler 1995

Issues in Workload-Driven Evaluation

- Choosing workloads
- Scaling workload and architecture
- Picking parameters and analyzing sensitivity
- Metrics, and presenting results

© Culler 1995