The Charm++ Programming Model

Chee Wai Lee

5/11/2010
Outline

• What is Charm++?

• An intelligent Charm++ runtime system.

• Large Charm++ applications.

• Performance Analysis of Charm++ programs.

• Where can you get Charm++?
What is Charm++?

• Object-based, message-driven model.
• Programmer envisions a collection of communicating objects.
What is Charm++? (2)

• Objects communicate through the invocation of Entry Methods.

```cpp
class Buyer {
    void getStuff(Seller s) {
        s.buys(this);
    }
    entry send(Item stuff) {
        // do things with stuff
    }
};
```

```cpp
class Seller {
    int i;
    Item stock[];
    entry buys(Seller who) {
        who.send(stock[i--]);
    }
};
```
What is Charm++? (3)

- Uses an adaptive runtime system to populate a parallel computer system with objects.
What is Charm++? (3)

• Under the hood: A **runtime scheduler** on each processor!
What is Charm++ (4)

1) Incoming message to a processor joins a queue

2) Execution Engine examines messages on the queue and decides which entry method to execute on which object.
Adaptive Overlap

• Overlapping of Communication with Computation.
• When one object runs out of work and waits, another object can be scheduled if work is available (on the queue).
• Works when there are more than 1 object per processor.

Object 2: Entry Method Y
Object 0: Entry Method Y

Processor is Idle
Object 2 runs out of work
Scheduler finds work for Object 0 on the message queue
Guiding Philosophies

• Processor Count Independence
  – Write code to use as many objects as necessary to solve the problem.
  – Preferably, use many more objects than processors (for Adaptive Overlap).

• Programmer Productivity
  – Try not to worry about what work goes where.
  – Allow the runtime system to deal with performance issues where possible.

TRUST THE RUNTIME, DUDE!!!
An Intelligent Runtime?

- What does it offer?
- What can it buy us?
Measurement-based Load Balancing Framework

• The Charm++ runtime system has a built-in per-object work/communication measurement framework.

• Load Balancing strategies uses the measurements to decide if:
  – There is load imbalance across the processor space.
  – Which objects need to be moved and to which processors.
Measurement-based Load Balancing Framework (cont)

• New load Balancing strategies can be written by application developers.
• Built-in strategies exist, examples include:
  – Greedy
  – Greedy with Communication
Measurement-based Load Balancing (cont)

Having a few heavily overloaded processors is bad news! Why?
Measurement-based Load Balancing (cont)

Having a few under-loaded processors is not necessarily bad! Why?
Measurement-based Load Balancing (cont)

• To adjust load, objects are serialized and migrated to target processors.

• Done infrequently (assuming the application satisfies the **Principle of Persistence**).

• Principle of Persistence:
  – that application behavior changes slowly over time (eg. Some molecular dynamics simulations); or
  – that application behavior changes abruptly but relatively infrequently (eg. Simulation of weather systems).
NAMD: A Production MD program

NAMD
• Fully featured program
• NIH-funded development
• Distributed free of charge (~5000 downloads so far)
• Binaries and source code
• Installed at NSF centers
• User training and support
• Large published simulations (e.g., aquaporin simulation featured in SC’02 keynote)
Object Based Parallelization for MD: Force Decomposition + Spatial Decomposition

- Create new objects:
  - One for each pair of neighbor cubes

- Now, we have many objects to load balance:
  - Each diamond can be assigned to any proc.
  - Number of diamonds in 3D = $-14 \cdot \text{Number of Patches}$
NAMD Parallelization using Charm++ : PME

These 30,000+ Virtual Processors (VPs) are mapped to real processors by charm runtime system.
Other Benefits of the Intelligent Runtime

• Automatic Fault Tolerance support with fast recovery
  – If one processor dies, objects from that dead processor gets restarted on another processor, possibly in an older state.
  – All other work not dependent on restarted objects can continue to do work while waiting for these objects to catch up.
Other Benefits of the Intelligent Runtime (cont)

• Faster Out-of-core processing
  – By examining the message queue, objects can be more effectively pre-fetched from disk into memory.
MPI vs Charm++: The Programming Model

• A matter of problem design. Modeling entities, tasks and communication.

• MPI
  – parallel entity: SPMD process;
  – task: function;
  – communication: message

• Charm++
  – parallel entity: object;
  – task: object entry method;
  – communication: message
MPI vs Charm++

• Whatever Charm++ can do, MPI can do. One of the underlying implementations of the Charm++ runtime uses MPI!

• With MPI, the programmer just has to get his/her hands (very) dirty in order to support the same dynamic/adaptive features.
Performance Analysis of Charm++ Applications

• Why? Runtime system is intelligent, not perfect!

• Lots of places for developer intervention:
  – Different load balancing strategies.
  – Placement of fixed-location objects.
  – Bad sequential algorithms.
  – Bad communication patterns.
Performance Event-Tracing in Charm++

- Default instrumentation is automatic.
- Typical events captured:
  - Arrival of a message triggers the execution of an entry method.
  - The end of an entry method triggers a return to the scheduler.
  - The sending of a message to another object.
Performance Analysis of Charm++

- Performance data can then be processed by some performance tool:
  - The Java tool **Projections** distributed with the Charm++ system.
  - **TAU**
Visualization of Charm++ Performance: Projections

Per-Processor (x-axis) utilization

Detailed timelines of processors (y-axis)
Visualization of Charm++ Performance: TAU

Performance correlation between two Charm++ events

TAU Scaling studies of NAMD on BG/P
Getting Charm++

• Visit http://charm.cs.uiuc.edu, where you can find:
  – Manuals
  – Tutorials
  – Research papers
  – Descriptions of various projects using/improving Charm++
  – Video presentations of talks (Annual Charm++ Workshops)