Exascale Applications

• Main challenge: variability
  – Hardware variation
    • Static/dynamic, heterogeneity, failures, power, etc.
  – Dynamic program behavior
    • AMR, particle movements, subscale simulations, ...

• To deal with these, we must seek:
  – Not full automation
  – Not full burden on the app-developers
  – But a good division of labor between the app-developer and system
Charm++

• Charm++ is a general-purpose object-oriented parallel programming system
  – Built on an adaptive runtime system

• Three principles that empower an adaptive runtime system:
  – Overdecomposition
  – Migratability
  – Asynchrony
Overdecomposition

- Decompose the work units & data units into many more pieces than execution units
  - Nodes/cores/…

- Not so hard: we do decomposition anyway
Migratability

• Allow these work and data units to be migratable at runtime
  – So the programmer or runtime can move them

• Consequences for users
  – Communication must be addressed to logical units with global names, not to processes
  – But this is a good thing

• Consequences for RTS
  – Naming and location management
Asynchrony

- We have multiple units on each processor
- They address each other via logical names
  - How do we schedule them?

- Message-driven execution:
  - Let the work-unit that happens to have data ("message") available for it execute next
  - Let the RTS select among ready work units
Charm++: Object-based overdecomposition

• Multiple indexed collections of C++ objects
  – Indices: multidimensional and/or sparse
• Programmer expresses communication between objects
  – Objects communicate via asynchronous remote method invocation
  – With no reference to processors: A[i].foo(…)

User view

System view
Message-driven Execution

Process 0

Scheduler

Message Queue

A[..].foo(…)

Process 1

Scheduler

Message Queue
Adaptive Runtime Systems

• Decomposing a program into a large number of migratable objects empowers the RTS to:
  – Map and migrate objects at will
  – Schedule tasks when they have work
  – Instrument computation and communication
    • Object A communicates $x$ bytes to B every iteration
  – Maintain historical data to track changes in application behavior
    • i.e. to trigger load balancing
Projections

• Performance visualization tool for Charm++
Fault Tolerance

• Basic Ideas:
  – Checkpoints are just migrations to storage
  – Underlying storage can be various things
  – Can be used in concert with load balancing

• Four approaches available:
  – Disk-based checkpoint/restart
  – In-memory double checkpoint w/ auto restart
  – Proactive object migration
  – Message-logging
Interoperability with MPI

- Implement new libraries/modules in the model that fits it best
  - Reuse existing libraries
  - Incremental adoption path
  - Already in production use for petascale apps: NAMD, OpenAtom, EpiSimdemics

(a) Time Division

(b) Space Division
Adaptive MPI

- MPI-2.2 implementation on top of Charm++
  - MPI ranks are lightweight, migratable user-level threads associated with Charm++ objects
**AMPI migration**

- AMPI can transparently migrate ranks
Adaptive MPI

- Application-independent features:
  - Over-decomposition via process virtualization
  - Automatic overlap of comm. & comp.
  - Dynamic load balancing
  - Fault tolerance

- Issue: global/static variables are shared by all ranks in the same OS process
  - But we have automated compiler tools for privatization
Near Future Plans

• Merging now:
  – Improved GPU manager
  – Job shrink–expand
  – Online performance autotuning
  – Fine-grained message aggregation

• Ongoing work:
  – AMPI compliance with MPI–3.1
  – Improved node–level threading/tasking library
  – OpenMP thread/task scheduling integration
Summary

• Charm++ is a scalable, adaptive runtime system for asynchronous parallel computing

• Many applications have been developed using it
  – NAMD, ChaNGa, EpiSimdemics, OpenAtom, …
  – Many mini-apps and third-party apps

• Lesson: adaptivity developed for apps is useful for addressing exascale challenges
  – Adaptivity to hardware and software factors

www.charmplusplus.org
Thank you
# Charm++ production apps

<table>
<thead>
<tr>
<th>Application</th>
<th>Domain</th>
<th>Previous parallelization</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMD</td>
<td>Classical MD</td>
<td>PVM</td>
<td>500k</td>
</tr>
<tr>
<td>ChaNGa</td>
<td>N-body gravity &amp; SPH</td>
<td>MPI</td>
<td>500k</td>
</tr>
<tr>
<td>EpiSimdemics</td>
<td>Agent-based epidemiology</td>
<td>MPI</td>
<td>500k</td>
</tr>
<tr>
<td>OpenAtom</td>
<td>Electronic Structure</td>
<td>MPI</td>
<td>128k</td>
</tr>
<tr>
<td>Spectre</td>
<td>Relativistic MHD</td>
<td></td>
<td>100k</td>
</tr>
<tr>
<td>FreeON/SpAMM</td>
<td>Quantum Chemistry</td>
<td>OpenMP</td>
<td>50k</td>
</tr>
<tr>
<td>Enzo-P/Cello</td>
<td>Astrophysics/Cosmology</td>
<td>MPI</td>
<td>32k</td>
</tr>
<tr>
<td>ROSS</td>
<td>PDES</td>
<td>MPI</td>
<td>16k</td>
</tr>
<tr>
<td>SDG</td>
<td>Elastodynamic fracture</td>
<td></td>
<td>10k</td>
</tr>
<tr>
<td>ADHydro</td>
<td>Systems Hydrology</td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Disney ClothSim</td>
<td>Textile &amp; rigid body dynamics</td>
<td>TBB</td>
<td>768</td>
</tr>
<tr>
<td>Particle Tracking</td>
<td>Velocimetry reconstruction</td>
<td></td>
<td>512</td>
</tr>
<tr>
<td>JetAlloc</td>
<td>Stochastic MIP optimization</td>
<td></td>
<td>480</td>
</tr>
</tbody>
</table>
AMPI codes (with no porting effort)

Mantevo 3.0
  CoMD 1.1
  HPCCG 1.0
  MiniFE 2.0
  MiniMD 2.0
  MiniXYCE 1.0

LLNL ASC Proxy Apps
  AMG 2013
  Kripke 1.1
  LULESH 2.0
  Lassen 1.0

LLNL Libraries
  HYPRE 2.10.1
  MFEM 3.0.1
  XBraild 1.1

Other apps
  SNAP (C) 1.01
  PENNANT 0.8
  PRK 2.16