Ensemble Dispatching on an IBM Blue Gene/L for a Bioinformatics Knowledge Environment

Presenter: Paul Marshall
University of Colorado at Boulder

Matthew Woitaszek, Henry Tufo, Rob Knight,
Daniel McDonald, Julia Goodrich, Jeremy Widmann

NCAR

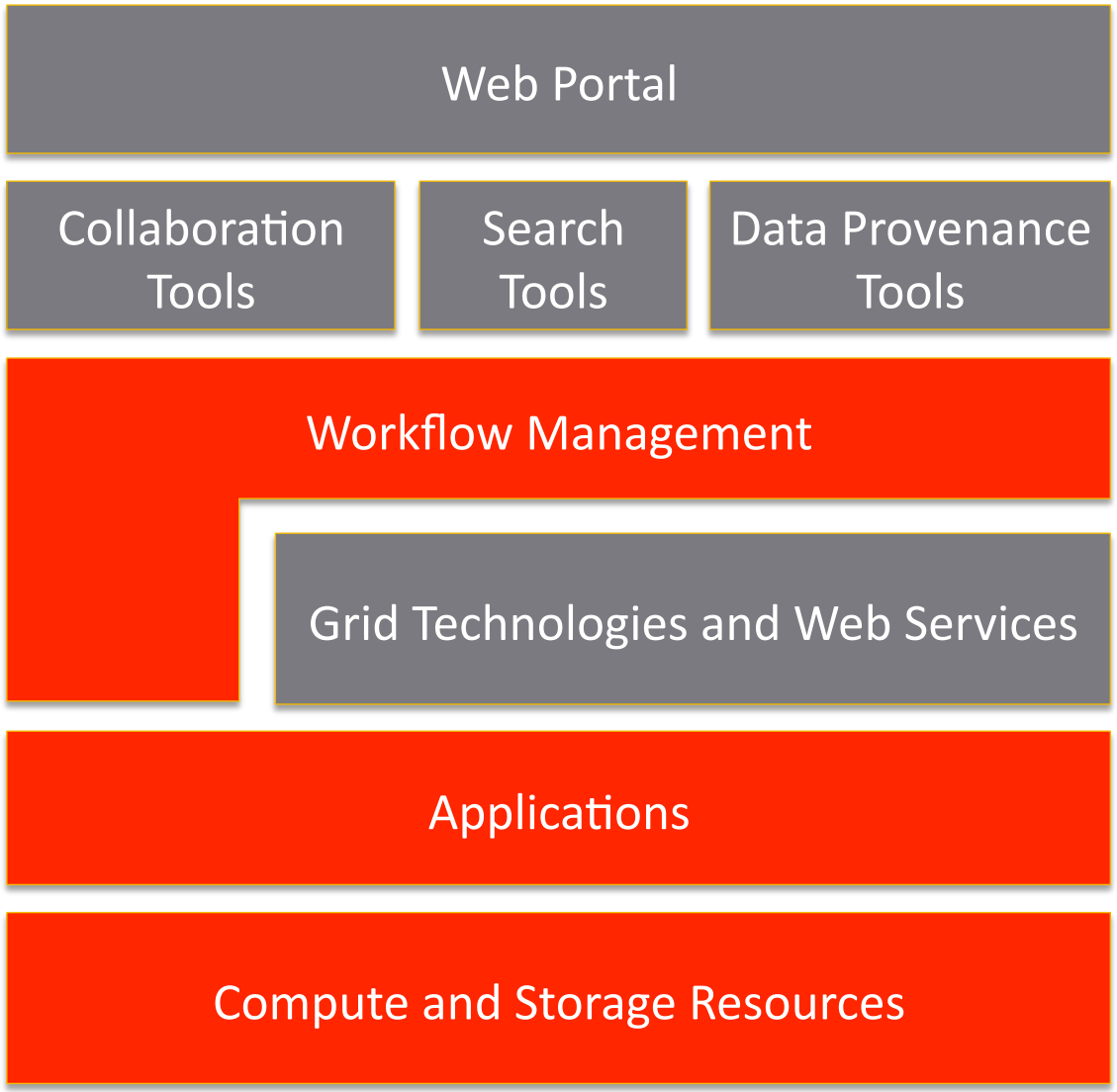
Overview

- Bioinformatics Knowledge Environment
- Workload characteristics
- High Throughput Computing (HTC) on the Blue Gene/L
- Implementation
- Results

Bioinformatics Knowledge Environment

- Encourage collaboration among structural biologists, environmental microbiologists, and evolutionary biologists
- Understanding the structure and functions of the biomolecules that are used to infer phylogeny
- Emerging gene sequencing technologies yield hundreds of thousands of sequences

Bioinformatics Knowledge Environment



Initial Applications

- FastTree
- Clearcut
- RAxML

Workload Characteristics

- Many small input files
  - 5KB up to a few hundred KB
  - To date we have processed over three million input sequences
- Many small output files
- Command line execution
- Parameter sweeps
- Runtimes:
  - RAxML: tens of minutes up to a few hours
  - Clearcut and FastTree: tens of seconds up to tens of minutes

Why Blue Gene/L?

- Why not Blue Gene/P?

- Why not a Linux cluster?
- Why not Condor?
- Why not the Cloud?

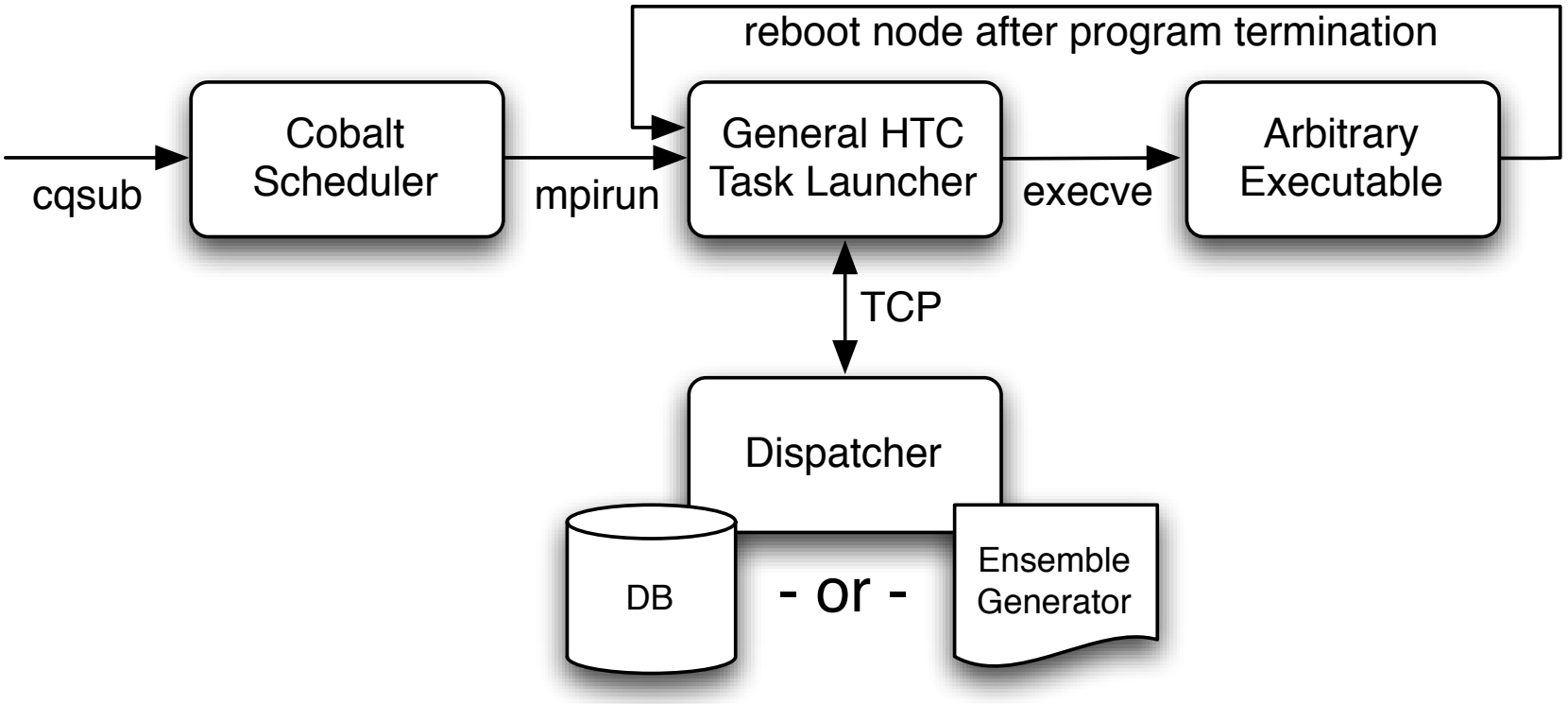
IBM Blue Gene/L

- Our primary large-scale computing system
- Frost: 8192-core Blue Gene/L at the National Center for Atmospheric Research (NCAR)
- Intended for massively parallel MPI applications
- Users allocate processors in 64-core partitions

HTC on the Blue Gene/L

- 64-core partitions reserved by a single user
- However, now individual cores can execute different executables
- But, no mechanism for dispatching tasks
  - Hardcoded launcher
  - execve
  - Reboot

HTC Task Dispatching



Problems with HTC Dispatcher

- Dedicated database server
- Over a million tasks added to the database at once
- Tasks dispatched to over 1000 cores
  - Query the database
  - Update records
- Database completely overwhelmed
  - Database tweaks (prefetching) didn’t help enough

Options

- More resources
- Modify the applications to bundle tasks
- Eliminate the database

Ensemble Dispatcher

- Eliminate the database
- Avoid tracking individual items in memory or a database
- Ensembles
  - Data set represented as files on disk
  - Generators that define the operations to be performed on individual data elements

Implementation

- Python ensemble dispatcher that runs on the front end nodes
- Ensemble dispatcher imports generators
- Launcher (written in C) executes on compute nodes
- Communication between dispatcher and launchers over TCP with a custom protocol

Generators

- Iterate through the data set
- Yield an operation to be performed on each data element
  - Executable
  - Parameters
  - Standard output file
  - Standard error file

Implementation of the Generators

- Generators are implemented as custom Python modules
  - Create one generator for each ensemble
  - Examine output directories for recovery

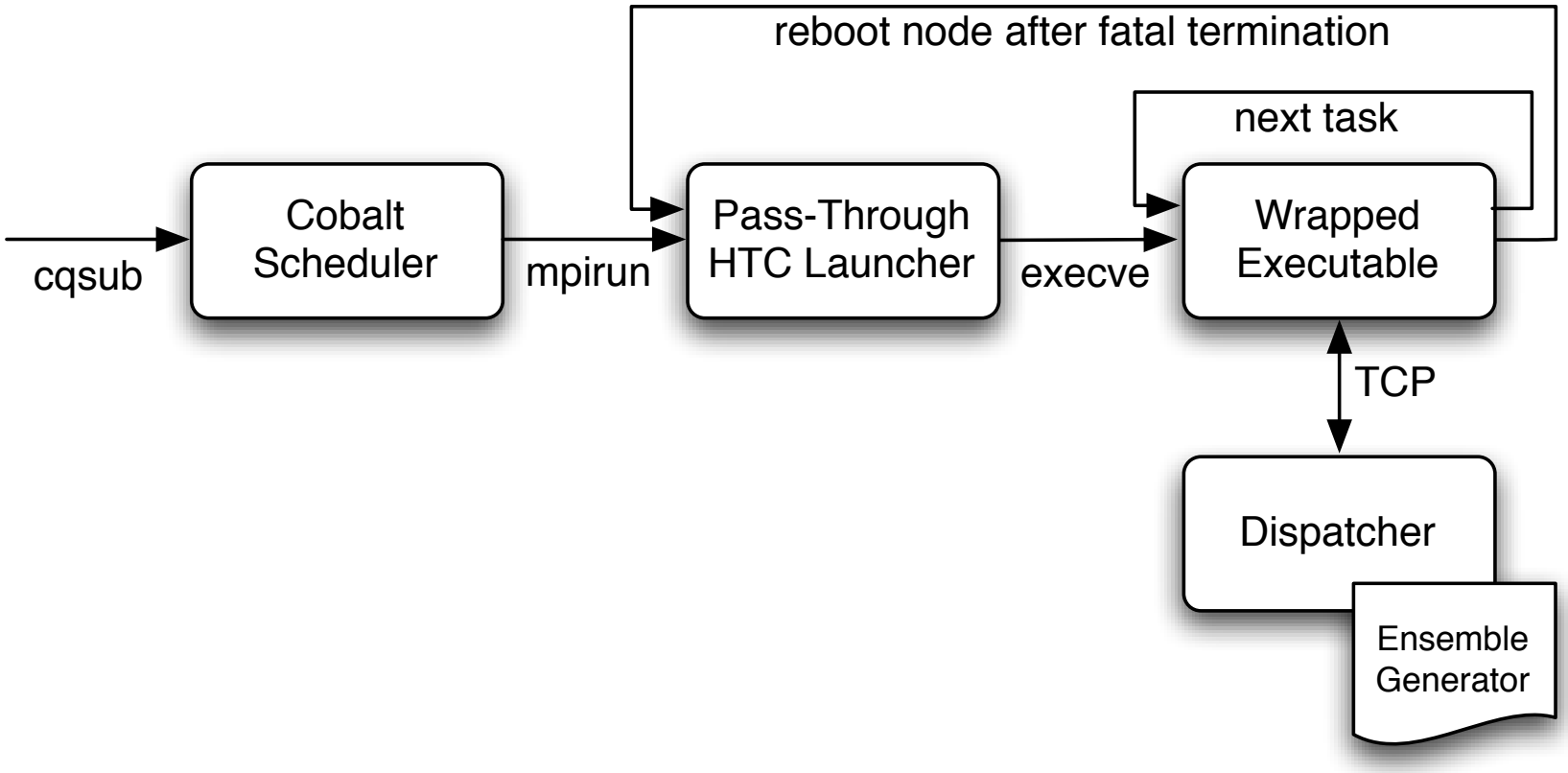
Let's try this again…

- Still slow
- Possibilities:
  - Ensemble dispatcher
  - Blue Gene/L
- Looking closer:
  - Node reboots between each task

Wrapped HTC Executables

- Eliminating node reboot overhead
- Include dispatcher integration in application executables
  - Wrap the program's main() function with code to contact the dispatcher
  - If program executes successfully it loops to receive another task

Ensemble Dispatching with a Wrapped Executable



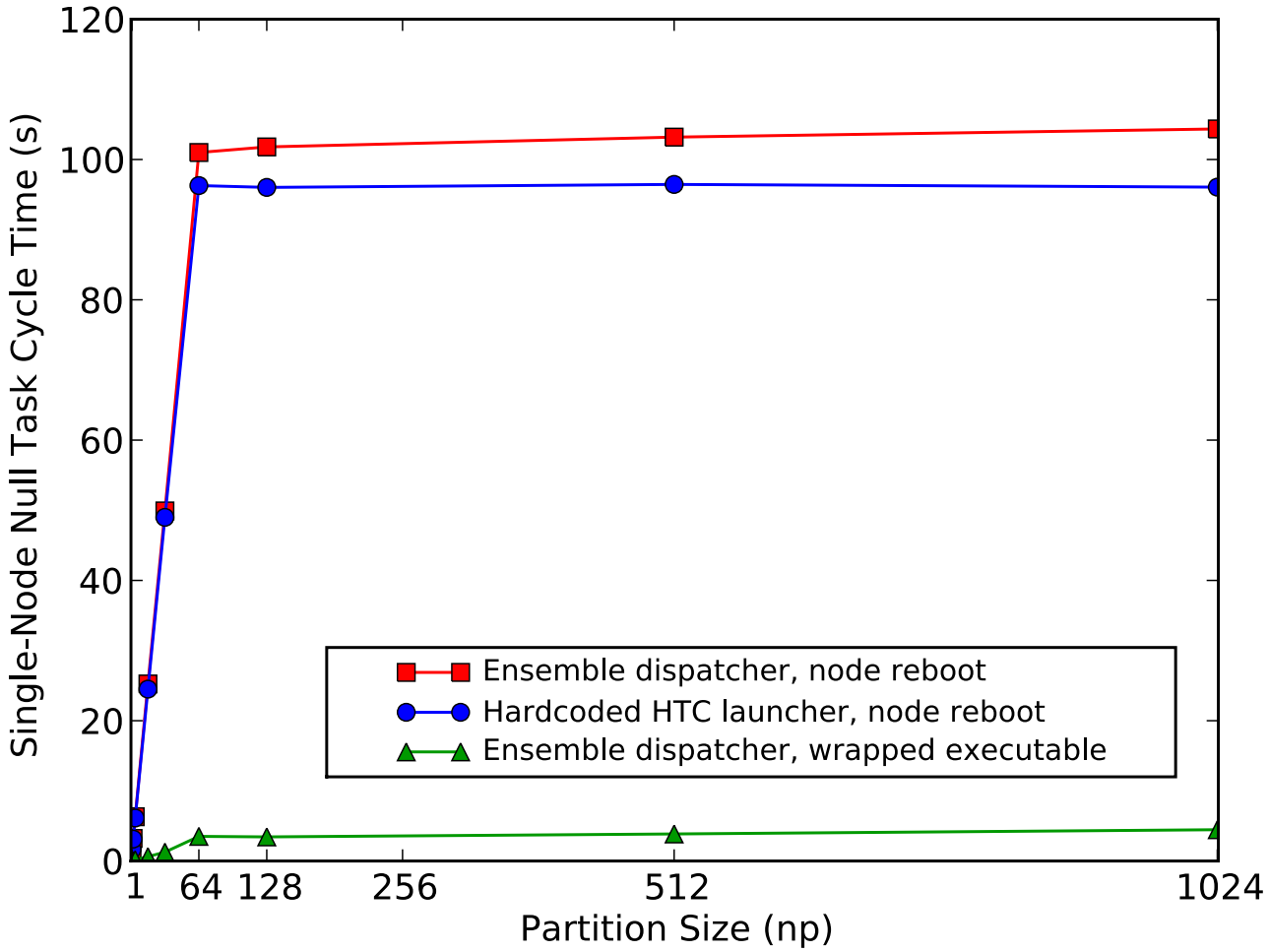
Concerns of Wrapped Executables

- Problematic if the application's programmers did not follow good memory management practices
- Global variables must be reinitialized properly
- Loss of generality

Blue Gene/L HTC Analysis

- Measure task cycle time
- Hardcoded null tasks (sleep 0 seconds)
- Execute from 1 core to 64 cores, then multiple 64 core partitions

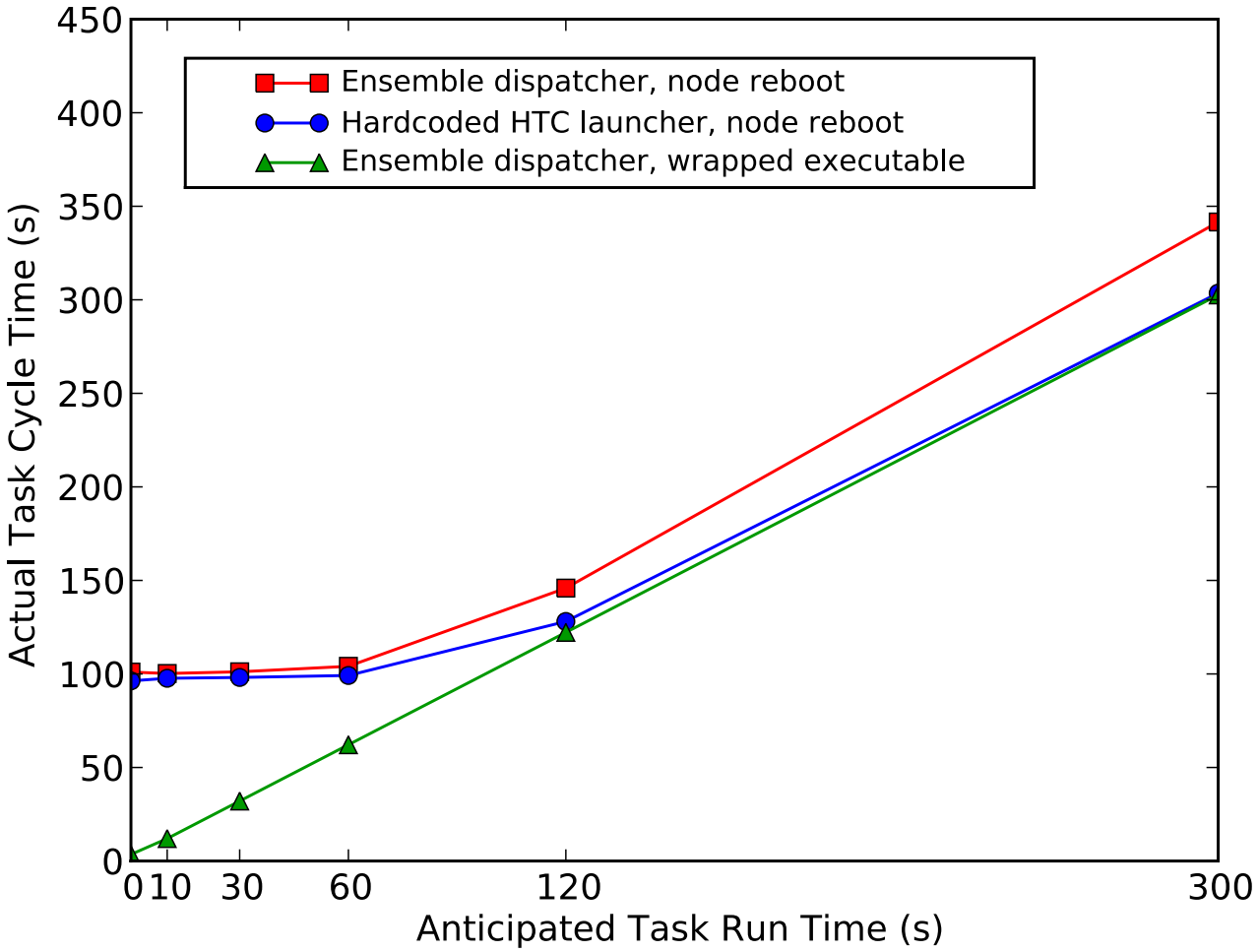
Blue Gene/L HTC Analysis



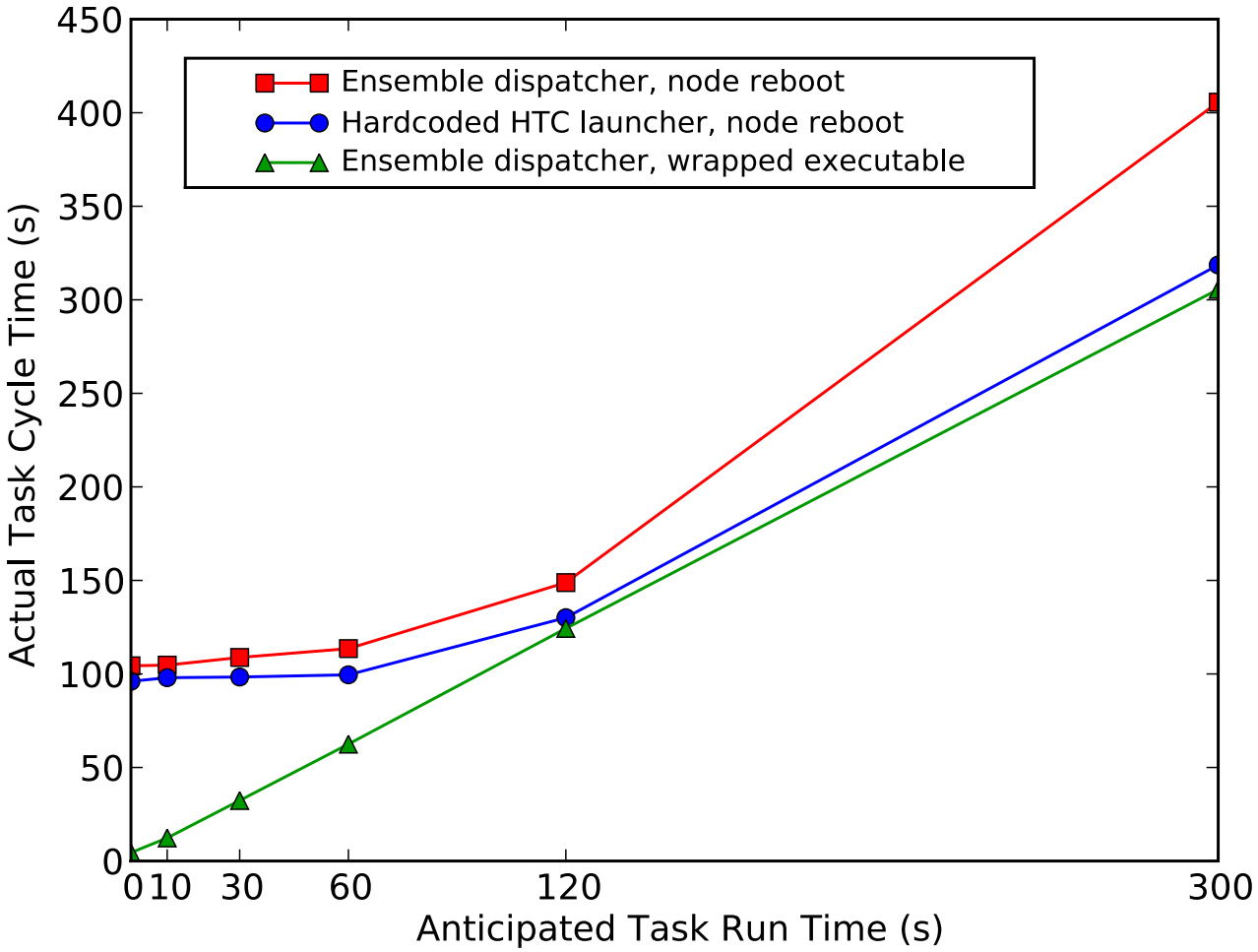
Blue Gene/L HTC Analysis

- Task cycle time increases as more processors are booted on a single 64-core partition
  - When a processor boots its kernel is transferred over the 100 Mbps JTAG service network
  - Each 64-core partition contains 1 service connection, nodes utilize it in a serial fashion

Task Dispatching and Launching Results 64-cores



Task Dispatching and Launching Results 1024-cores



Bioinformatics Test Case

- Wrapped Clearcut executable
- 12,600 unique input files
- 256 nodes with 512 cores

- Original Clearcut: 2694 seconds
- Wrapped Clearcut: 844 seconds

Conclusions

- The Blue Gene/L is able to process our bioinformatics workload efficiently
- For long running tasks there is no need to wrap the executable
- Short running tasks should be wrapped
- Ensemble dispatcher maintains minimal state in RAM and can efficiently resume execution by examining output directories

Future Work

- Deploy on more resources
  - Grid resources (TeraGrid)
  - Cloud resources (Amazon, Nimbus Clouds)
- Current work is only a preliminary step towards the larger bioinformatics knowledge environment
  - Integrate the ensemble dispatcher into the knowledge environment
  - Create and manage dynamically
  - Data annotation and provenance

Thank You

Paul Marshall

University of Colorado at Boulder

paul.marshall@colorado.edu