Nexus

A common substrate for cluster computing

Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ion Stoica, Scott Shenker
Problem

Rapid innovation in cluster computing frameworks

No single framework optimal for all applications

Running multiple frameworks in a single cluster
Solution

**Nexus** is a resource manager over which frameworks like Hadoop can be written

» Nexus multiplexes resources between frameworks

» Frameworks control job execution
Implications

Users can pick best framework for each app
Specialized frameworks, not one-size-fits-all
I only want to use Hadoop

Nexus is a better way to manage Hadoop

Hadoop master is complex, hard to scale and make robust

Multiple Hadoop instances/versions at same time
Outline

Beyond MapReduce and Dryad
Nexus Architecture
Implementation
Philosophy
Beyond MapReduce & Dryad
1. Iterative Jobs

Many machine learning jobs are of the form:

```plaintext
p = random();
while (p not converged) {
    p = f(p, dataset);
}
```
2. Nested Parallelism

Recursion (quicksort), maps within maps

Difficult in MapReduce/Dryad, possible with NESL model
3. Irregular Parallelism

Sometimes, we don’t know computation graph
  » Branch-and-bound search
  » Exploring moves in chess
  » Ray tracing

Hard to hack into MapReduce/Dryad, easy with work-stealing programming model (Cilk)
4. Existing Parallel Apps

Parallel build (distcc)

Parallel unit test (Selenium Grid)

Web servers (!)
Nexus Architecture
Hadoop

App

Hadoop master

Hadoop slave

Hadoop slave

Hadoop slave

Hadoop slave

Task

Task

Task
Nexus

- App
- Hadoop scheduler
- Nexus master
  - Nexus slave
    - Hadoop executor
      - task
  - Nexus slave
    - Hadoop executor
      - task
  - Nexus slave
    - Hadoop executor
      - task
Scheduler API

---

- Framework scheduler
  - slot_offer(slot)
  - accept_offer(task)
  - reject_offer()
  - status(task, status)

---

Nexus master

---
Executor API

Nexus slave

start_task(task)

kill_task(task)

status(task, status)

Framework executor
Analysis

Frequency of slot offers
  \( t = \) average task length (e.g. 60s)
  \( r = \) # replicas (e.g. 3)
  \( s = \) slots per node (e.g. 8)

Avg slot offer wait time = \( \frac{t}{rs} \) (e.g. 2.5s)
Analysis

Right of first refusal
Provides “code locality”

Grab and hold
Avg co-located slot offer wait time = \( t / s \)
Implementation
Implementation Status

Simple
2000 lines of C++

Scalable
500 slaves on EC2

Frameworks
Preliminary port of Hadoop, and specialized LR framework
LR Job Comparison

![Bar chart showing comparison of Hadoop, Hadoop on Nexus, and Nexus across iterations.](image-url)
Philosophy
Microkernel
   »Make reliable component as small as possible

Exokernel
   »Give maximal control to frameworks

IP model
   »Narrow waist over which diverse frameworks can run
Questions