Datacenter Management with Apache Mesos

mesos.apache.org

@ApacheMesos

Benjamin Hindman – @benh
I’ve got tons of data ...
... more everyday!
That must be why they call it a data center.
I’d love to answer some questions with the help of my data!
I think I’ll try Hadoop.
your datacenter
+ Hadoop
happy?
Not exactly ...
... Hadoop is a big hammer, but not everything is a nail!
I’ve got some iterative algorithms, I want to try Spark!
datacenter management
datacenter management
static partitioning
Oh noes! Spark wants to read and write data to HDFS!
Hadoop ...

- Hadoop
  - (map/reduce)
  - HDFS (distributed file system)
HDFS
HDFS
Could we just give Spark its own HDFS cluster too?
HDFS
HDFS
HDFS
HDFS

tee incoming data
(2 copies)
HDFS

tee incoming data
(2 copies)

periodic copy/sync
That sounds annoying ... let’s not do that. Can we do any better though?
HDFS
HDFS
HDFS
happy now?
No! We’ve decided to start doing real time computation with Storm ...
datacenter management
datacenter management
happy now!?
Not really ... during the day I'd rather give more machines to Spark but at night I'd rather give more machines to Hadoop!
datacenter management
datacenter management
datacenter management
datacenter management
And failures require more datacenter management!
datacenter management
datacenter management
datacenter management
I don’t want to deal with this!
the datacenter ...

rather than think about the datacenter like this ...
... is a computer

think about it like this ...
datacenter computer

applications
resources
filesystem
mesos

- hadoop
- Spark
- HDFS

Layered Structure:
- Applications
- Kernel
- Resources
- Filesystem
Okay, so how does it work?
Step 1: HDFS
Step 2: Mesos

run a “master” (or multiple for high availability)
Step 2: Mesos

run “slaves” on the rest of the machines
Step 3: Frameworks
Step 3: Frameworks

- Hadoop
- HDFS
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks

- Hadoop
- Spark
- HDFS
Step 3: Frameworks

- Hadoop
- Spark

HDFS
Step 3: Frameworks
Step 3: Frameworks
Step 3: Frameworks
Step 4: Profit
Step 4: Profit (utilize)

just one big pool of resources, utilize single machines more fully!
Step 4: Profit (utilize)


Step 4: Profit (utilize)

- Hadoop
- Spark
- HDFS
Step 4: Profit (utilize)
Step 4: Profit (utilize)
Step 4: Profit (utilize)
$tep 4: Profit (statistical multiplexing)
Step 4: Profit
(statistical multiplexing)
Step 4: Profit
(statistical multiplexing)
Step 4: Profit (statistical multiplexing)
$\textit{Step 4: Profit (statistical multiplexing)}$
Step 4: Profit (statistical multiplexing)

reduces CapEx and OpEx!
Step 4: Profit (statistical multiplexing)

reduces latency!
Step 4:

- Profit (statistical multiplexing)
Step 4: Profit (failures)
$\text{Step 4: Profit (failures)}$
Step 4: Profit (failures)
This sounds pretty good!
Other than Hadoop, Spark, and Storm, what else can I run on Mesos?
frameworks

- Hadoop (github.com/mesos/hadoop)
- Spark (github.com/mesos/spark)
- DPark (github.com/douban/dpark)
- Storm (github.com/nathanmarz/storm)
- Chronos (github.com/airbnb/chronos)
- MPICH2 (in mesos git repository)
- Aurora (proposed for Apache incubator)
What about XYZ?
port an existing framework

strategy: write a “wrapper” which launches existing components on mesos

~100 lines of code to write a wrapper (the more lines, the more you can take advantage of elasticity or other mesos features)

see src/examples/ in mesos repository
write a new framework!

as a “kernel”, mesos provides a lot of primitives that make writing a new framework relatively easy

primitives: extracted commonality across existing distributed systems/frameworks (launching tasks, doing failure detection, etc) ... why re-implement them each time!?
case study: chronos

distributed cron with dependencies

developed at airbnb

~3k lines of Scala!

distributed, highly available, and fault tolerant without any network programming!

http://github.com/airbnb/chronos
Hmm ... if Mesos gives me a *datacenter computer* ... can I run stuff other than analytics?
case study: aurora

run $N$ instances of my server, somewhere, forever
(where server == arbitrary command line)

developed at Twitter

runs hundreds of production services, including ads!

recently proposed for Apache Incubator!
aurora
aurora
aurora
aurora
But what about resource isolation!? I don’t want my end users to have to wait for our website to load because of resource contention!
resource isolation

Linux control groups (cgroups)

CPU (upper and lower bounds)
memory
network I/O (traffic controller)
filesystem (lvm, in progress)
conclusions

datacenter management is a pain
conclusions

mesos makes running frameworks on your datacenter easier as well as increasing utilization and performance while reducing CapEx and OpEx!
conclusions

rather than build your next distributed system from scratch, consider using mesos
conclusions

you can share your datacenter between analytics and online services!
Questions?

mesos.apache.org

@ApacheMesos