Software Session Notes:

What are the main differences and commonalities between the HPC and BDA requirements/technologies/working-assumptions in this area?

Overview of slides/findings from last meeting.

Data differences:

- Not really not repeatable -- not a feature of Big Data by itself
- Sensor data is often very noisy -- more noisy than simulation data
- In BD applications often you can combine data that is not high quality in itself but when you combine it you get a lot of signal whereas in simulation often you operate on data of known quality -- if a big MPI job fails the whole job fails -- in BD there is an assumption of noise and that the noise can be tolerated
- BD is often volatile -- e.g., produced by sensors that come online and go offline, whereas in EC size is typically well understood and predictable
- In BD addition to input coming to instruments you also have stored data (e.g., a baseline.
- Surprised by the last bullet saying unstructured -- because even data produced by sensors is going to be structured
- Unstructured data == web data, scraped -- but working with scientific partners things are typically quite structured
- If you take "scientific" out of this column that opens it up a bit wider
- Any data could be scientific -- if we talk about social networks, and sensors, etc. -- opens the definition of data quite a bit wider
- In BD we have a wider definition of data than EC
- Scientific data is data that is used for scientific purposes
- Nobody wants to defend unstructured versus semi-structured division
- Differences between scientific and non-scientific data is in how it is used produced/consumed, if you process it for scientific purposes -- the fact that it is not only open (and inspectable) but also evolving process
- E.g., search versus patient diagnostics -- a lot of similarities here
- In google you can't inspect the process -- consumers are not scientists don't have flexibility
- EC versus BD is just a question of ratios: a lot of flops versus big input big output.
- Argument is that EC is a very narrow range of applications
- Division implied in the current column design is driven by the need to account for the emergent data patterns (otherwise we've been doing BD for a long time)
- BD is not a well defined concept
- One division to think about the storage requirements as being a 1st class citizen instead of addition to large flop count == decouple data from HPC system
- Reinforcement of "shared and curated' versus "often private"

- The intent is to separate what our current software stack does not handle well -- and in other words the driving force for change
- Start from HPC -- now more data coming in because of emergent applications and pattern --
- The output can be less reliable and this can be tolerated
- Dynamicity, volatility, and availability requirements -- HPC systems are too static
- Archivable -- needs to be preserved in a more structured way
- Machine learning, unstructured data mining becomes a more interesting technique -- there
- Exploration more focused on domain independent data patterns than domain dependent methods -- BD should not be based on size but the new things
- Folks who are doing HPC applications are moving to "BD" piece by piece
- Convergence -- software stack in BD is slow, not scaling -- they presumably need to borrow some techniques from HPC
- BD people don't understand how to process data
- Productivity versus performance
- Satoshi: in HPC we have a choice -- in BD there isn't
- Can't combine super optimized HPC stuff with super inefficient BD derivative stuff
- In academic science capital expense is expensive, labor is cheap -- in industry it is the opposite
- Summary of last few points: there is value in sharing technology across communities
- In HPC we need to have a little bit more focus on productivity
- Growing performance is hard in general -- there is opportunity for convergence
- Productivity versus performance
- General feeling that we would have done it (== BD) better!
- Adaptation -- no HPC system does well (independently of the cost of labor)
- Emphasis on utilization in HPC (batch) versus control over response time or some other SLA (on-demand)
- A lot of the time we could started out from different assumptions -- then new requirements came in -- but those assumptions are not hardwired in the design of EC systems
- Communication between nodes -- BD -- matters less of r are heterogenous
- whereas BG -- built on the assumption of total uniformity
- Early cloud solution was provided on BG/P
- How should convergence be structured? What do we take from each community? Incorporating or converging
- Efficiency in EC is a big concern whereas in BD qualitative issues (like reliability and such things) are more of a driver
- Batch queue systems were built to optimize utilization
- You can't have BOTH response time and good utilization
- The tradeoffs should not be hardwired into the infrastructure -- utilization versus response time -- infrastructure should be able to accommodate both

- No agreement with fine-grained versus large bulk division -- large bulk could be on Big data side -- change to storage access -- it is storage access should be fine grained not data access -- and EC does not do fine-grain storage well
- In EC we view data as living on a SAN versus in BD the computation lives on a WAN
- Line 4, we broadly disagree: some of the BD models are standardized, e.g., MapReduce -- not sure what the customized ones are (maybe the intent was to convey control over environment)
- In BD we need rich meta-data (down to who can process this bit of data) that is not so often met with on the EC side
- IN BD the vision of concurrency is very simplistic -- concurrency models are very simplistic -- data model is disconnected from the concurrency model -- another trade-off -- in EC models are potentially too concurrency driven
- MapReduce is very old and batch oriented and no longer a new model -- Berkeley's spark, some of the newer model now support transactions, support consistency trade-offs,
- On EC side we don't have streaming technology
- BG doesn't support dynamic process creation (the software doesn't) -- hardware and software are alike impacted by focus on EC
- MPI is not necessarily the right model, we need data flow in HPC as well
- Today people don't consider task-based to be HPC -- but it could/should be
- Line 4: customized is not the right word but data algebra versus compute algebra -- but also qualitative, sophisticated e.g., based on consistency trade-offs --- but unsophisticated data model BSP
- Bottom line in EC -- not true that they almost never but that they are never allowed to
- What does it mean that they are resilient to fault?
- BD resilience == stateless + on-demand
- Replication is the cornerstone of resilience in BD so it is resilience at a price
- Wrong: that HPC systems have totally different hardware to BD -- the differences are mainly in software -- Kate doesn't think she agrees with that
-

Identify Scenarios that one does well and the other not:
- BD is coming towards EC, discovering IB and RDMA -- a lot of opportunities -- BD has features and needs to optimize them
- But on the other hand they ask for functionality that we do not have
- Missing in EC
    - we need something more

____

Are there common needs/problems/interfaces could serve as the basis (or as stepping stones) along a path to (some reasonable level of) infrastructure and application convergence?

- performance

- Are there interdomain testbeds that combine BDA and HPC workflows in ways that could help uncover pathways toward convergence?
- What is/are the technology or new research that may be a game changer?
- What action would be your number one priority to be taken rapidly to ensure success of the converge of Extreme computing and Big Data infrastructures?
- What action would be your number one priority to be taken rapidly to ensure the emergence of efficient Extreme computing and Big Data applications?
- How would you measure the success of the BDEC initiative?

Software Session Notes
Leads: Frank, Satoshi
Scribe: Kate

Q1 Cleaned up (a little bit)

What are the main differences and commonalities between the HPC and BDA requirements/technologies/working-assumptions in this area?

Discussion based on the overview of Pete's slides/findings from last meeting.

- BD not repeatable: Not really not repeatable -- not a feature of Big Data by itself -- you could repeat it if you record it

- Noisy: Sensor data is often very noisy -- more noisy than simulation data
- In BD applications often you can combine data that is not high quality in itself but when you combine it you can raise its quality whereas in simulation often you operate on data of known quality

- Resilience: if a big MPI job fails the whole job fails -- in BD there is an assumption of failure and all types of noisiness can be tolerated

- Volatility in BD: BD is often volatile -- e.g., produced by sensors that come online and go offline, requests to BD services come and go whereas in EC data volume is typically well understood and predictable and thus the resource needs are more controlled -- BD responds to a pattern that *relies* on a flexible platform -- on-demand is not choice but a requirement

- Unstructured -- surprised by the last bullet saying unstructured -- because even data produced by sensors is going to be structured
- Unstructured data == web data, scraped -- but working with scientific partners things are typically quite structured
- Nobody wants to defend unstructured versus semi-structured division
- Maybe unstructured == noisy in the sense that even data that are not well annotated when taken together can add up to interesting things

- What is scientific data?
- With data from social networks, sensors, etc. definition between scientific and non-scientific data is becoming blurred
- Scientific data is data that is used for scientific purposes
- Difference in the context: differences between scientific and non-scientific data is in how it is used produced/consumed, if you process it for scientific purposes -- the fact that it is not only open (and inspectable) but also subject to an evolving process
- E.g., search versus patient diagnostics -- a lot of similarities here
- In google you can't inspect the process -- consumers are not scientists don't have flexibility

- What is Big Data?
- EC versus BD is just a question of ratios: a lot of flops versus big input/big output
- Argument is that EC is a very narrow range of applications
- Division implied in the current column design is driven by the need to account for the emergent data patterns (otherwise we've been doing BD for a long time)
- Maybe New Data instead of Big Data?

- One division to think about the storage requirements as being a 1st class citizen instead of addition to large flop count == decouple data from HPC system

- Reinforcement of "shared and curated' versus "often private"
- Archivable -- needs to be preserved in a more structured way

- Dynamicity, volatility, and availability requirements: HPC systems are too static

- Processing
- More focused on general data processing techniques -- machine learning, unstructured data mining becomes a more interesting techniques -

- Exploration more focused on domain independent data patterns than domain dependent methods

- BD people don't understand how to process data
- Software stack in BD is slow, not scaling -- they presumably need to borrow some techniques from HPC
- Can't combine super optimized HPC stuff with super inefficient BD derivative stuff
- General feeling that we would have done it (== BD) better!

- Focus on productivity versus performance
- In academic science capital expense is expensive, labor is cheap -- in industry it is the opposite
- In HPC we need to have a little bit more focus on productivity

- Summary of last few points: there is value in sharing technology across communities
- Growing performance is hard in general -- there is opportunity for convergence

- Adaptation -- no HPC system does well (independently of the cost of labor)
- Emphasis on utilization in HPC (batch, when computers were expensive) versus control over response time or some other SLA (on-demand)
- Batch queue systems were built to optimize utilization
- You can't have BOTH response time and good utilization

- Communication between nodes -- BD -- matters less of r are heterogenous
- whereas BG -- built on the assumption of total uniformity
- Early cloud solution was provided on BG/P

- Efficiency in EC is a big concern whereas in BD qualitative issues (like reliability and such things) are more of a driver

- We started out from different assumptions -- then new requirements came in -- but those assumptions are now hardwired in the design of EC systems
- The tradeoffs should not be hardwired into the infrastructure -- utilization versus response time -- infrastructure should be able to accommodate both
- No agreement with fine-grained versus large bulk division -- large bulk could be on Big data side -- change to storage access -- it is storage access should be fine grained not data access -- and EC does not do fine-grain storage well
- In EC we view data as living on a SAN versus in BD the computation lives on a WAN

- EC standardized models versus BD customized: we broadly disagree: some of the BD models are standardized, e.g., MapReduce -- not sure what the customized ones are (maybe the intent was to convey control over environment)

- MapReduce is very old and batch oriented and no longer a new model -- Berkeley's spark, some of the newer model now support transactions, support consistency trade-offs,
- In BD we need rich meta-data (down to who can process this bit of data) that is not so often met with on the EC side
- IN BD the vision of concurrency is very simplistic -- concurrency models are very simplistic -- data model is disconnected from the concurrency model -- another trade-off -- in EC models are potentially too concurrency driven
- On EC side we don't have streaming technology
- Customized is not the right word but data algebra versus compute algebra -- but also qualitative, sophisticated e.g., based on consistency trade-offs --- but unsophisticated data model BSP
- Cultural difference: BD community seeks performance via qualitative trade-offs whereas EC community tends to fix the model and focus on raw performance optimization

- What does it mean that they are resilient to fault?
- BD resilience == stateless + on-demand
- Replication is the cornerstone of resilience in BD so it is resilience at a price
- Wrong: that HPC systems do not have totally different hardware to BD -- the differences are mainly in software -- Kate doesn't think she agrees with that

---

Software Session 01/30

Are there common needs/problems/interfaces that could serve as the basis (or as stepping stones) along a path to (some reasonable level of) infrastructure and application convergence?

Motivation for convergence:

- BD has a lot to learn from HPC, can leverage the experience, etc.
- And vice versa e.g., resource management -- a while ago a batch scheduler used to be OK but now as more complexity in usage modes emerges we need more sophisticated modes of resource management
- HPC software lacks certain functionalities that BD focused on
- WHy can't we throw out HPC stack and adopt BD stack?

List aspects of software stack and list where there are gaps:

- Resource management: batch scheduler on-demand availability (utilization versus flexibility), multi-aspect holistic scheduling of data and computation
- OS level -- motivation for virtualization/containers -- needed for isolation, control over environment, also isolation in terms of performance/QoS, fine-grain control, policy-driven resource management (control versus performance), Argo&Hobbes
- Security -- in HPC world coarse grain security (firewall on site level) in BD world fine-grained, closely targeted security (flexibility versus complexity), includes network
- Performance tools -- instrumentation/measurement/analysis of relevant -- HPC focus on compute side, BD focus on storage side -- we need holistic performance analysis for the whole workflow
- Programming models -- task-based models -- data and control need to be a part of the model can we move to task-based models
    - Stream-based programming models are not well supported in HPC
    - HPC programming models lack support for data models
- Resilience -- inflexible model intolerant of failure versus model based on redundancy -- can we make HPC resilience models more flexible (what Franck is doing)
- Workflow management -- more constraint-driven optimization in HPC workflows, more meta-data to represent provenance
- File system -- BD is database-oriented HPC is not -- add this aspect to HPC, convergence via object storage (but since there is no data model
- Network -- isolation/performance isolation/control/QoS -- SDN making a dent
- Runtime -- embodiment of everything above

Are there interdomain testbeds that combine BDA and HPC workflows in ways that could help uncover pathways toward convergence?

- Chameleon
- Grid5000
- Amazon
- CloudLab
- MOC
- JetStream (production, disciplinary science)
- Intrigger
- FED4Fire
- Analytics workbench

What is/are the technology or new research that may be a game changer?

- A demonstration -- software definition of the entire system so that same hardware can be used for HPC or Big Data at the same time
- Advances in optics and novel memory (and software to accommodate that)
- Improved performance/features of cloud technologies (more HPC awareness)
- Availability of exascale services in the cloud
- BD jobs requiring HPC properties such as high bw interconnect
- End of Moore's Law

What action would be your number one priority to be taken rapidly to ensure success of the converge of Extreme computing and Big Data infrastructures?

- Throw money on the problem (1 billion Euro anyone?) -- research funding calls focusing on "stepping stones"
- Mandate this as part of your system procurement (co-design effort -- after some problems were solved)
- More awareness of Big Data needs in funding infrastructures
- Documents: common vocabulary, definitions, classification and subclassification of Big Data, functional specification and a convergence architecture, what is the software stack? Ensure that we are speaking the same language and talking about the same thing
- Killer App grand challenge (disqualify Google alt. people are only allowed to win once)
- More workshop and forums, cross-pollination -- conferences (IEEE BigData, VLDB, SIGMOD), journal special issues, convergence workshop, facebook page…. ;-)
- Have workshops associated with conferences
- Work with software providers to provide features on a case by case basis
    - Containers or virtualization (lightweight) plus efficient communication
    - Resource management
- Solve incentive problems
- Success stories and community leaders: we have adopted it and it works

What action would be your number one priority to be taken rapidly to ensure the emergence of efficient Extreme computing and Big Data applications?

- The software infrastructure should be able to sustain the performance requirements of existing HPC application
- Not sacrifice performance (metric == maximize scientific discovery, time to solution)
- Have a cross-disciplinary workshop of software community and application community to determine specific priorities (co-design)

- We need traces, workloads, data, etc. that would enable us to explore this research space -- mini apps, mini data (mini Big Data, even), benchmarks, development of methodology for this area, BDEC 500, anyone?
- Leverage “melting pots” of software and application communities like SC

How would you measure the success of the BDEC initiative?

- Documents and methodology as defined earlier plus a document to list the key problems to address to drive community research (as measured by e.g., citations of the report)
- The formation of a community
- Suggest specific funding calls
- Long term success metrics: adoption by companies/centers and customers/users on both sides of the house (EC, BD)