Azure MapReduce

Thilina Gunarathne
Salsa group, Indiana University
Agenda

• Recap of Azure Cloud Services
• Recap of MapReduce
• Azure MapReduce Architecture
• Pairwise distance alignment implementation
• Next steps
Cloud Computing

• On demand computational services over web
  – Backed by massive commercial infrastructures giving economies of scale
  – Spiky compute needs of the scientists
• Horizontal scaling with no additional cost
  – Increased throughput
• Cloud infrastructure services
  – Storage, messaging, tabular storage
  – Cloud oriented services guarantees
  – Virtually unlimited scalability
• Future seems to be CLOUDY!!!
Azure Platform

• Windows Azure Compute
  – .net platform as a service
  – Worker roles & web roles

• Azure Storage
  – Blobs
  – Queues
  – Table

• Development SDK, fabric and storage
MapReduce

- Automatic parallelization & distribution
- Fault-tolerant
- Provides status and monitoring tools
- Clean abstraction for programmers
  - map (in_key, in_value) ->
    (out_key, intermediate_value) list
  - reduce (out_key, intermediate_value list) ->
    out_value list
Motivation

- Currently no parallel programming framework on Azure
  - No MPI, No Dryad
- Well known, easy to use programming model
- Cloud nodes are not as reliable as conventional cluster nodes
Azure MapReduce Concepts

• Take advantage of the cloud services
  – Distributed services, Unlimited scalability
  – Backed by industrial strength data centers and technologies
• Decentralized control
• Dynamically scale up/down
• Eventual consistency
• Large latencies
  – Coarser grained map tasks
• Global queue based scheduling
1. Client driver loads the map & reduce tasks to the queues
2. Map workers retrieve map tasks from the queue
3. Map workers download data from the Blob storage and start processing.
4. Reduce workers pick the tasks from the queue and start monitoring the reduce task tables.
5. Finished map tasks upload the results to Blob storage. Add entries to the respective reduce task tables.
6. Reduce tasks download the intermediate data products
7. Start reducing when all the map tasks are finished and when a reduce task is finished downloading the intermediate data products.
Azure MapReduce Architecture

- Client API and driver
- Map tasks
- Reduce tasks
- Intermediate data transfer
- Monitoring
- Configurations
Fault tolerance

• Use the visibility timeout of the queues
  – Currently maximum is 3 hours
  – Delete the message from the queue only after everything is successful
  – Execution, upload, update status
• Tasks will rerun when timeout happens
  – Ensures eventual completion
  – Intermediate data are persisted in blob storage
  – Retry up to 3 times
• Many retries in service invocations
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MapReduce</td>
<td>DAG execution, Extensible to MapReduce and other patterns</td>
<td>Iterative MapReduce</td>
<td>MapReduce-- will extend to Iterative MapReduce</td>
<td></td>
</tr>
<tr>
<td>Data Handling</td>
<td>HDFS (Hadoop Distributed File System)</td>
<td>Shared Directories &amp; local disks</td>
<td>Local disks and data management tools</td>
<td>Azure Blob Storage</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Data Locality; Rack aware, Dynamic task scheduling through global queue</td>
<td>Data locality; Network topology based run time graph optimizations; Static task partitions</td>
<td>Data Locality; Static task partitions</td>
<td>Dynamic task scheduling through global queue</td>
</tr>
<tr>
<td>Failure Handling</td>
<td>Re-execution of failed tasks; Duplicate execution of slow tasks</td>
<td>Re-execution of failed tasks; Duplicate execution of slow tasks</td>
<td>Re-execution of Iterations</td>
<td>Re-execution of failed tasks; Duplicate execution of slow tasks</td>
</tr>
<tr>
<td>Environment</td>
<td>Linux Clusters, Amazon Elastic Map Reduce on EC2</td>
<td>Windows HPCS cluster</td>
<td>Linux Cluster EC2</td>
<td>Window Azure Compute, Windows Azure Local Development Fabric</td>
</tr>
<tr>
<td>Intermediate data transfer</td>
<td>File, Http</td>
<td>File, TCP pipes, shared-memory FIFOs</td>
<td>Publish/Subscribe messaging</td>
<td>Files, TCP</td>
</tr>
</tbody>
</table>
Why Azure Services

• Virtually unlimited scalable distributed services
• No need to install software stacks
  – In fact you can’t 😊
  – Eg: NaradaBrokering, HDFS, Database
• Zero maintenance
  – Let the platform take care of you
• Availability guarantees
API

- ProcessMapRed(jobid, container, params, numReduceTasks, storageAccount, mapQName, reduceQName, List mapTasks)
- Map(key, value, programArgs, Dictionary outputCollector)
- Reduce(key, List values, programArgs, Dictionary outputCollector)
Develop applications using Azure MapReduce

- Local debugging using Azure development fabric
- DistributedCache
  - Bundle with Azure Package
- Compile in release mode before creating the package.
- Deploy using Azure web interface
- Errors logged to a Azure Table
SWG Pairwise Distance Alignment

• SmithWaterman-GOTOH
• Pairwise sequence alignment
  – Align each sequence with all the other sequences
### Application architecture

#### Block decomposition

<table>
<thead>
<tr>
<th></th>
<th>1 (1-100)</th>
<th>2 (101-200)</th>
<th>3 (201-300)</th>
<th>4 (301-400)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M1</td>
<td>M2</td>
<td>\textit{from M6}</td>
<td>M3</td>
</tr>
<tr>
<td>2</td>
<td>\textit{from M2}</td>
<td>M4</td>
<td>M5</td>
<td>\textit{from M9}</td>
</tr>
<tr>
<td>3</td>
<td>M6</td>
<td>\textit{from M5}</td>
<td>M7</td>
<td>M8</td>
</tr>
<tr>
<td>4</td>
<td>\textit{from M3}</td>
<td>M9</td>
<td>\textit{from M8}</td>
<td>M10</td>
</tr>
</tbody>
</table>

- Reduce 1
- Reduce 2
- Reduce 3
- Reduce 4
AzureMR SWG Performance
10k Sequences

Number of Azure Small Instances vs Execution Time (s)
AzureMR SWG Performance
10k Sequences

Alignment Time (ms)

Number of Azure Small Instances

Time Per Alignment Per Instance
AzureMR SWG Performance on Different Instance Types

![Graph showing execution time for different instance types: Small, Medium, Large, ExtraLarge]
AzureMR SWG Performance on Different Data Sizes

![Graph showing the relationship between the number of sequences and the time for an actual alignment in milliseconds. The time decreases as the number of sequences increases.](image-url)
Next Steps

• In the works
  – Monitoring web interface
  – Alternative intermediate data communication mechanisms
  – Public release

• Future plans
  – AzureTwister
    • Iterative MapReduce
Thanks!!

• Questions? 😊
References


• Cloudmapreduce, http://sites.google.com/site/huanliu/cloudmapreduce.pdf


• M. Isard, M. Budiu, Y. Yu et al., "Dryad: Distributed data-parallel programs from sequential building blocks." pp. 59-72.
Acknowledgments

• Prof. Geoffrey Fox, Dr. Judy Qiu and the Salsa group
• Dr. Ying Chen and Alex De Luca from IBM Almaden Research Center
• Virtual School Organizers