CloudClustering

Toward a scalable machine learning toolkit for Windows Azure

Ankur Dave
XCG Intern
Microsoft Research

Mentors:
Roger Barga
Wei Lu

August 12, 2010
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. CloudClustering: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. **CloudClustering:** Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
About me

**EDUCATION**

Interlake High School

UNIVERSITY OF CALIFORNIA, BERKELEY

**WORK**

Berkman
The Berkman Center for Internet & Society at Harvard University

DreamBox Learning

JAIN FOUNDATION INC

CH2M HILL

**HOBBIES**

SET
The Family Game of Visual Perception

HALO
REACH

Microsoft Visual Studio
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. **CloudClustering:** Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
My team

Microsoft Research

Cloud Computing Futures

To create novel data center solutions, designs must be based on comprehensive optimization of all attributes, rather than gradually accruing incremental changes based on current technologies and best practices. The Cloud Computing Futures team is tasked to invent on a large scale. Our goal is to reduce data center costs by four-fold or greater, including power consumption, while accelerating deployment and increasing adaptability and resilience to failures.

Azure Research Engagement

The Azure Research Engagement project aims to change the paradigm for scholarly and scientific research by extending the power of the computer into the cloud. We build the components of cloud technology and work with researchers in the field on projects that push the frontier of client and cloud computing.

Dan Reed  Dennis Gannon  Jaliya Ekanayake  Jared Jackson  Nelson Araujo  Roger Barga  Wei Lu
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. **CloudClustering:** Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Internship: Domain

**Goals:**
- Build a scalable clustering algorithm on Azure
- Explore clustering and the cloud by reading papers
- Meet a variety of researchers at MSR

**Week 1:** Ramp-up; architecture planning

**Week 2:** Building CloudClustering base impl.

**Week 3:** Building CloudClustering base impl.

**Week 4:** Multicore parallelism with PLINQ

**Week 5:** Performance testing on Azure fabric

**Week 6:** Multicore w/threads; data affinity

**Week 7:** Presentation and report-out
1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. **CloudClustering**: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Windows Azure

Exploring

for data-intensive research
Windows Azure

Block upload -> Block commit -> Block Blob

Instance

Queue

Table

PartitionKey | RowKey | Col1 | Col2
---|---|---|---

---

Cloud Computing: Azure blob
7-week internship
- Cloud Computing: Azure batch
- Cloud Computing: Azure HDInsight
- Cloud Computing: Azure HDInsight: MapReduce
- Cloud Computing: Azure HDInsight: Spark
- Future work
- Evaluation of Windows Azure for data intensive problems
- Relationship research
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. CloudClustering: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Clustering
Clustering
Clustering

**k-means**

- Minimizes the *within-cluster sum of squares*:

\[
\sum_{i=1}^{k} \sum_{p \in C_i} \| p - \mu_i \|^2
\]

For each cluster

For each point in the cluster

Distance from point to centroid
Clustering

Target clustering
Clustering

1. Initialization

$k = 3$
Clustering

2. Assign Points to Centroids #1
Clustering

3. Recalculate Centroids #1
Clustering

4. Assign Points to Centroids #2
Clustering

5. Recalculate Centroids #2
Clustering

Stopping Condition
Related Work
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. CloudClustering: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
CloudClustering: Algorithm

Conventional (Serial) $k$-means
**CloudClustering: Algorithm**

**Parallel $k$-means**

1. **Start parallel run**
2. **Initialization**
3. **Partition work**
4. **Assign points to centroids**
5. **Return partial sums**
6. **Assign points to centroids**
7. **Return partial sums**
8. **Assign points to centroids**
9. **Return partial sums**
10. **Recalculate centroids**

**Decision diamond**:
- **Total # of points reassigned > 0**
- **No**: Return points and centroids
- **Yes**: Continue the algorithm

**Flowchart**:
- **Server**
- **Worker**
CloudClustering: Implementation

Server control queue

Task result queue

Server

Worker queue 1

Worker queue 2

Worker queue 3

Worker 1

Worker 2

Worker 3

Points blob
(x, y, closestCentroidID)

Centroids blob
(ID, x, y)
CloudClustering: Implementation

Server control queue

Task result queue

Points blob

(x, y, closestCentroidID)

Centroids blob

(ID, x, y)

Worker queue 1

Worker queue 2

Worker queue 3

Worker 1

Worker 2

Worker 3

Start parallel run

Initialization

Partition work

Assign points to centroids

Return partial sums

Assign points to centroids

Return partial sums

Assign points to centroids

Return partial sums

Recalculate centroids

Total # of points reassigned > 0

No

Return points and centroids

Yes
CloudClustering: Leveraging the cloud

- Multicore Parallelism
- Data Affinity
- Efficient Blob Concatenation
- Dynamic Scalability
CloudClustering: Multicore Parallelism

Thread-based data flow

Input blob

Multiple IO channels

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

Multiple IO channels

Output blob

PLINQ data flow

Input blob

Single IO channel

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

Assign Point To Nearest Centroid

foreach

Single IO channel

Output blob
Thread-based

```csharp
private void ProcessPoints()
{
    CloudBlockBlob pointsBlob = AzureHelper.GetBlob(task.Points);
    // Do the mapping and write the new blob
    int numThreads = Environment.ProcessorCount;
    var partitions = Task.Parallel.For(0, numThreads, threadIdx =>
    {
        {
            // Process the points
            foreach (var point in stream)
            {
                // Assign the point to the nearest centroid
                Guid oldCentroidID = point.CentroidID;
                int closestCentroidIndex = centroids.MinIndex(centroid => Point.Distance(point, centroid));
                Guid newCentroidID = centroids[closestCentroidIndex].ID;
                // Update the point sums
                if (oldCentroidID != newCentroidID)
                {
                    pointsChangedPerThread[threadID]++;
                }
                // Update the point sum
                if (pointSumsPerCentroidPerThread[threadID, closestCentroidIndex] == null)
                {
                    pointsSumsPerCentroidPerThread[threadID, closestCentroidIndex] = new PointsProcessedData();
                }
                pointSumsPerCentroidPerThread[threadID, closestCentroidIndex].PartialPointSum += point;
                pointsSumsPerCentroidPerThread[threadID, closestCentroidIndex].NumPointsProcessed++;
            }
            // Collect the block IDs from write stream
            blockIDsPerThread = writeStream.BlockList.ToArray();
            // Combine the per-thread block lists and write the full block list to a blob. Then include that as part of TaskResult.Lists<
            foreach (string[] blockIDsFromThread in blockIDsPerThread)
            {
                blockIDs.AddRange(blockIDsFromThread);
            }
            using (Stream stream = blockIDsBlob.OpenWrite())
            {
                BinaryFormatter bf = new BinaryFormatter();
                bf.Serialize(stream, blockIDs);
            }
            TaskResult.PointsProcessedDataByCentroid = new Dictionary<Guid, PointsProcessedData>();
            for (int i = 0; i < task.K; ++i)
            {
                Guid centroidID = centroids[i].ID;
                TaskResult.PointsProcessedDataByCentroid[centroidID] = new PointsProcessedData();
            }
            for (int j = 0; j < numThreads; ++j)
            {
                if (pointSumsPerCentroidPerThread[j, i] != null)
                {
                    TaskResult.PointsProcessedDataByCentroid[centroidID].PartialPointSum += pointSumsPerCentroidPerThread[j, i].PartialPointSum;
                    TaskResult.PointsProcessedDataByCentroid[centroidID].NumPointsProcessed += pointSumsPerCentroidPerThread[j, i].NumPointsProcessed;
                }
            }
            // Total up the per-thread NumPointsChanged
            TaskResult.NumPointsChanged = 0;
            foreach (int threadPointsChanged in pointsChangedPerThread)
            {
                TaskResult.NumPointsChanged += threadPointsChanged;
            }
        }
    });
}
```

PLINQ

```csharp
private void ProcessPoints()
{
    CloudBlockBlob pointsBlob = AzureHelper.GetBlob(task.Points);
    // Do the mapping and write the new blob
    var assignedPoints = stream.AsParallel().SelectAssignClusterPointToNearestCentroid();
    using (ObjectBlockWriter<ClusterPoint> writeStream = new ObjectBlockWriter<ClusterPoint>(pointsBlob, point =>
        point.ToByteArraay(), ClusterPoint.Size),
        TaskResult.PointsProcessedDataByCentroid = new Dictionary<Guid, PointsProcessedData>();
    while (Pipeline execution -- see http://msdn.microsoft.com/en-us/magazine/cc163329.aspx)
    {
        // Write the point to the new blob
        writeStream.Write(point); // Update the number of points changed
        if (result.PointsWasChanged)
        {
            TaskResult.PointsWasChanged++;
        }
        // Add to the appropriate centroid group
        if (TaskResult.PointsProcessedDataByCentroid.ContainsKey(point.Point.CentroidID))
        {
            TaskResult.PointsProcessedDataByCentroid[point.Point.CentroidID].NumPointsProcessed++;
        }
        // Send the block list as part of TaskResult
        writeStream.FlushBlock();
        TaskResult.PointsBlockList = writeStream.BlockList;
    }
    private ClusterPointProcessingResult AssignClusterPointToNearestCentroid(ClusterPoint clusterPoint)
    {
        ClusterPoint result = new ClusterPoint(clusterPoint);
        result.CentroidID = centroids.MinElement(centroid => Point.Distance(clusterPoint, centroid)).ID;
        return new ClusterPointProcessingResult
        {
            Point = result,
            PointsWasChanged = clusterPoint.CentroidID != result.CentroidID
        };
    }
```
CloudClustering: Multicore Parallelism

Scale-up: Speed up for varying instance size, PLINQ vs. Threads

Scale-up: Cost for varying instance size, PLINQ vs. Threads
Scale-out: Time and speedup for varying instance counts

- Time for 1 iteration (minutes)
- Speedup vs. 1 instance

# of instances

- Time for 1 iteration
- Speedup compared to 1 instance
CloudClustering: Data Affinity

Points blob

Worker

Read ~20 MB/s

Cache ~50 MB/s

Local disk
CloudClustering: Data Affinity

Typical Azure architecture

- Server
- Central task queue
- Worker pool

Architecture with data affinity

- Server
- Worker queue
- Cache
CloudClustering: Data Affinity

Time for varying instance size, No Affinity vs. Affinity (Threads, 4 instances)

Time to complete 1 iteration (minutes)

Instance size (# of cores)

- Threads time
- Threads+Affinity time
CloudClustering: Efficient Blob Concatenation

Server control queue

Task result queue

Points blob

Centroids blob

Worker 1

Worker 2

Worker 3

Start parallel run → Initialization → Partition work → Assign points to centroids → Return partial sums → Recalculate centroids

Total # of points reassigned > 0

No

Yes

Return points and centroids
CloudClustering: Efficient Blob Concatenation
CloudClustering: Dynamic Scalability
CloudClustering: Demo
CloudClustering: Demo
1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. CloudClustering: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Azure is an **appropriate level of abstraction** for data-intensive algorithms like $k$-means.
Windows Azure: Potential Problem Areas

– On the cloud, **cost** scales directly with **usage**
  – Sub-linear speedups are not good enough!

– For data-intensive algorithms, **data affinity** gives great performance... but there’s a **tradeoff**
  – Dynamic scaling is more complex
  – Fault-tolerance is even harder

– Performance test to find configuration **sweet spots**
Agenda

1. Me
2. Cloud Computing Futures team
3. 7-week internship
4. Windows Azure
5. Clustering research
6. Related work
7. **CloudClustering**: Algorithm, implementation, and optimizations
8. Benchmarks
10. Future work
Future work

• A compromise between worker pools and data affinity that retains scalability and fault-tolerance
  – Buddy system

• Improved caching using blocks

• Fundamental improvements to the $k$-means algorithm
  – More efficient stopping condition
  – “Lazy” processing that eliminates synchronization barriers

• Further optimizations to multicore parallelism
Ankur Dave

ankurdave@gmail.com
http://ankurdave.com