HPCHadoop: MapReduce on Cray X-series

Scott Michael
Research Analytics
Indiana University

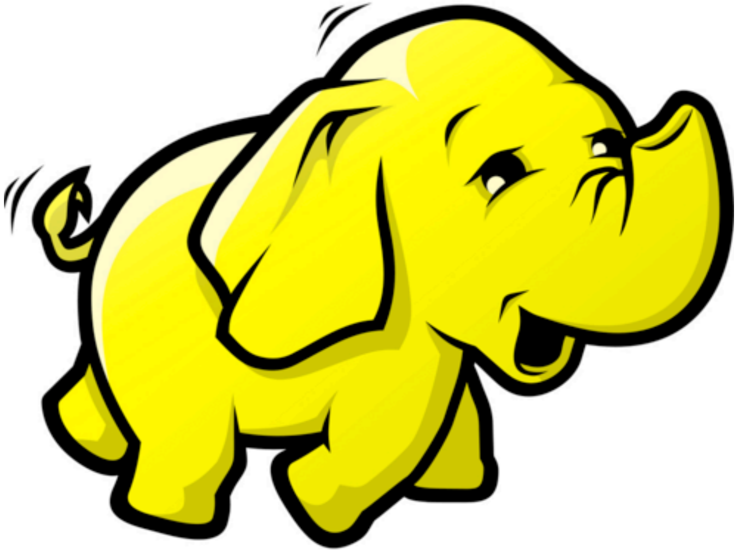
Cray User Group Meeting
May 7, 2014

Outline

- **Motivation & Design of HPCHadoop**
- HPCHadoop demo
- Benchmarking Methodology
- Benchmark Results
- Future Work

Why Hadoop on a Cray?

- Many users have heard about Big Data and Hadoop and want to try it out
- Some users already have Hadoop code
- Being a relatively simple framework, Hadoop can lower the barrier to entry for distributed computing
- At IU departmental resources can be scarce, and HPC resources are "free" to faculty



Why Hadoop on a Cray?

MPI/OpenMP

MapReduce/Hadoop

Hadoop is all Java, right?

Hadoop is all Java, right?

- The Hadoop framework is Java based
- But Map and Reduce functions can be written in any language and streamed to the framework via Hadoop streaming
- For certain types of data reduction and analysis Hadoop can be a good fit
  - Astronomical image analysis
  - Medical image analysis
  - Genome analysis

Java just runs everywhere, right?

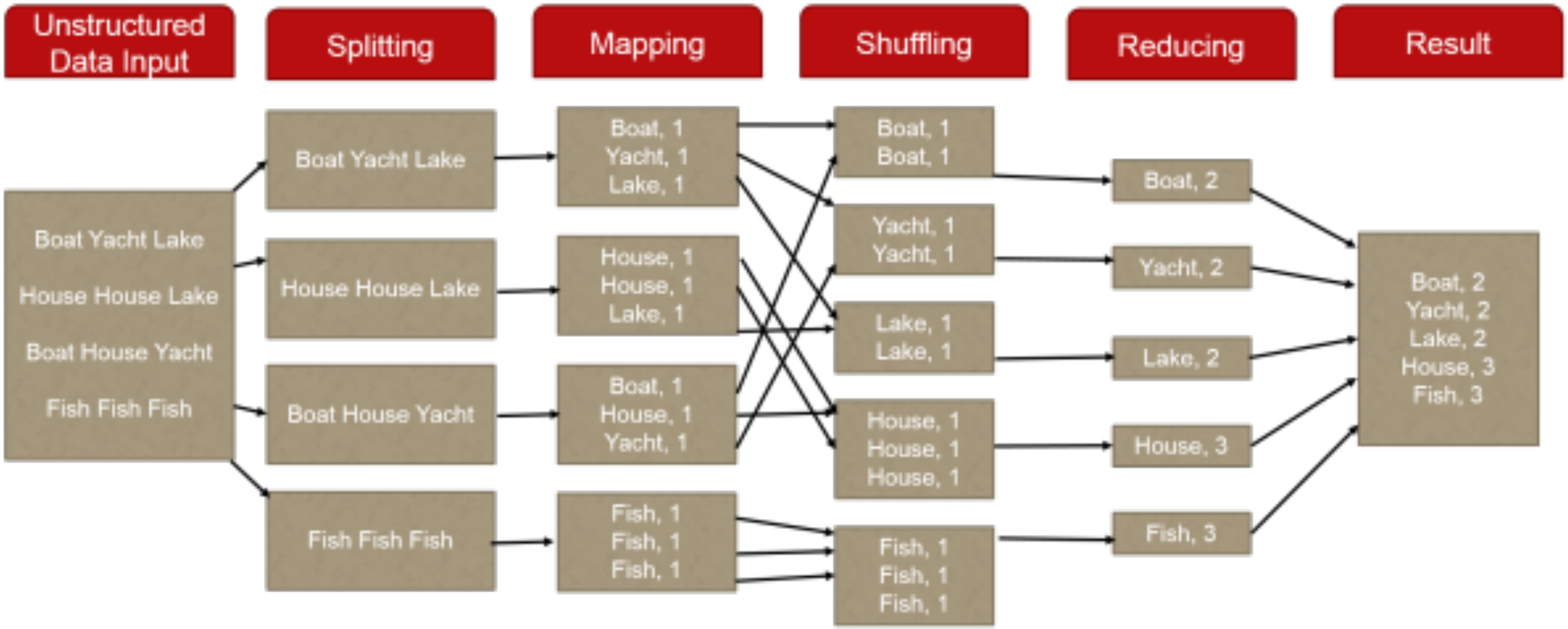
- There are two major challenges in deploying Hadoop on a traditional HPC resource
  - Shared scheduling
  - Hadoop's "shared nothing" architecture
- The framework has to address these issues *and* be easy to configure and run

Java just runs everywhere, right?

- Hadoop is generally deployed across an entire cluster that doesn’t change or only changes infrequently
- HPCHadoop takes information from the scheduler, configures and launches a Hadoop instance on your nodes
- HDFS can be instantiated on node local disks, or
- HDFS can be set up on a shared file system

- Motivation & Design of HPCHadoop
- **HPCHadoop demo**
- Benchmarking Methodology
- Benchmark Results
- Future Work

Wordcount Example



RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

Just two functions

$$map: \quad (k_1, v_1) \rightarrow list(k_2, v_2)$$

```
public void map(LongWritable key, Text value,
OutputCollector<Text, IntWritable> output, Reporter
reporter) throws IOException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
      word.set(tokenizer.nextToken());
      output.collect(word, one);
    }
   }
```

Just two functions

$$reduce: \quad (k_2, list(v_2)) \rightarrow list(k_3, v_3)$$

```
public static class Reduce extends MapReduceBase
implements Reducer<Text, IntWritable, Text, IntWritable>
{
 public void reduce(Text key, Iterator<IntWritable>
values, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
          sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
      }
    }
```

To the Terminal…

- Three configuration files need to be modified
  - conf/env.sh
  - conf/hadoop_commands.sh
  - PBS script
- Set inputs and outputs
- Set up a directory for HDFS

- Motivation & Design of HPCHadoop
- HPCHadoop demo
- **Benchmarking Methodology**
- Benchmark Results
- Future Work

Many Choices in Benchmarks

- There are many "standard" Hadoop benchmarks, but there is a lot of parameter space to explore
- We settled on the Intel Hadoop Benchmarking Suite called HiBench https://github.com/intel-hadoop/hibench
- Benefits & Drawbacks
  - Standard Suite gets you many benchmarks
  - Framework can get in the way and insists on HDFS for some benchmarks

Intel HiBench Suite

- HiBench gives a broad sampling of potential Hadoop workloads by including: Bayes, DFSIOE, Kmeans, Nutchindex, Pagerank, Terasort, and Wordcount
- The HiBench framework is relatively easy to set up and run, though it has a very large number of tunables
- We decided at minimum to optimize the numbers of mappers/ reducers and input data size
  - Mapper/reducer ratio -- 4:3
  - Input data size -- 2.5x default data size

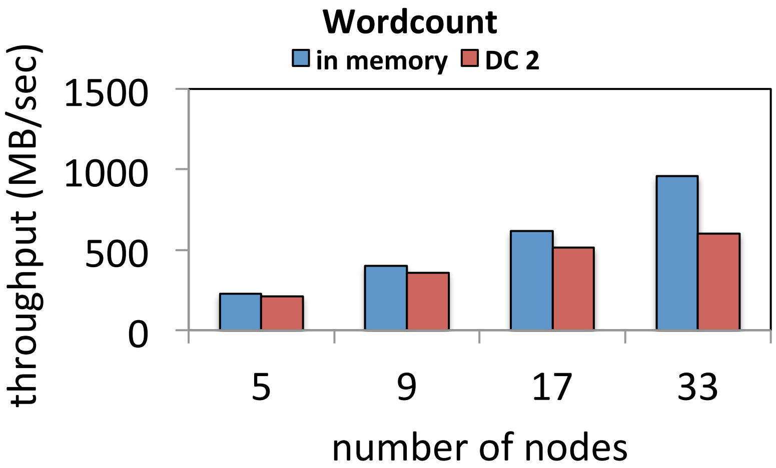
Benchmark Hardware

- Big Red II
  - Cray XE6/XK7
  - 32 cores/node (XE6)
  - 64 GB mem/node (XE6)
- Quarry
  - Intel based gigbit cluster
  - 8 cores/node
  - 16 GB mem/node
- Data Capacitor II
  - 5 PB Lustre filesystem

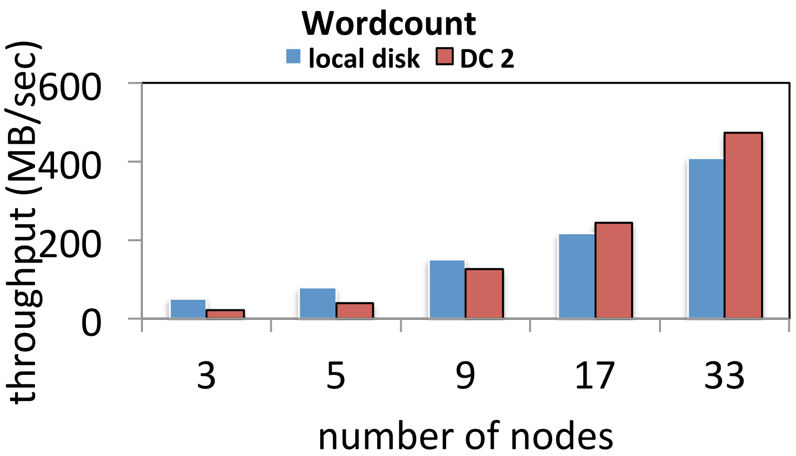
- Motivation & Design of HPCHadoop
- HPCHadoop demo
- Benchmarking Methodology
- **Benchmark Results**
- Future Work

Some Results

- In all 20 runs of HiBench across the two machines; 160 individual benchmark results



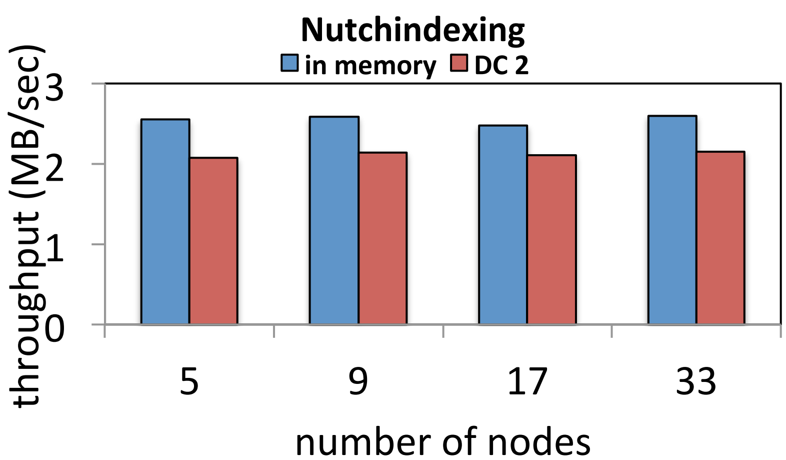
Big Red II Wordcount scalability



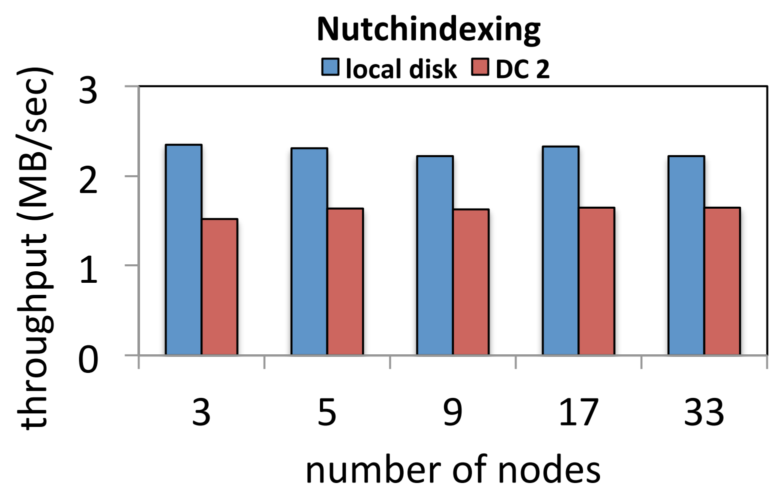
Quarry Wordcount scalability

Not everything scales well

- This could be due to the algorithm, or simply require further optimization



Big Red II Nutchindexing scalability



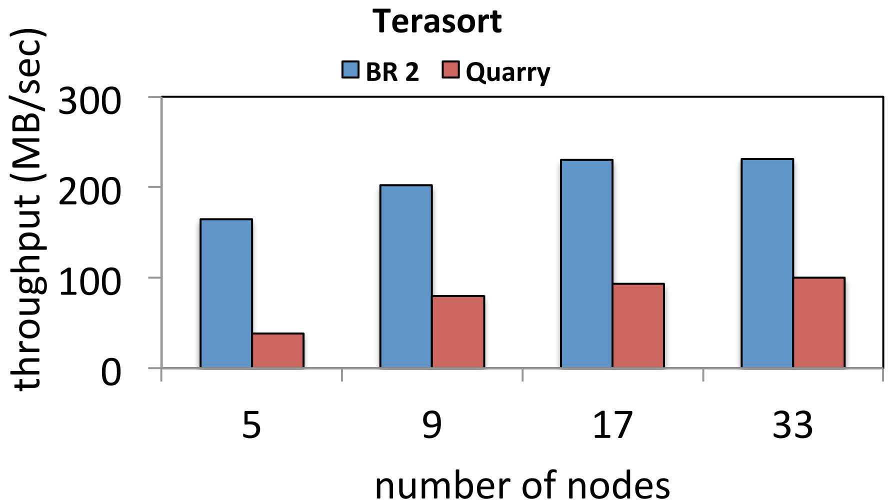
Quarry Nutchindexing scalability

RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

X-series can perform well

- Comparing Cray X-series to gigabit connected cluster for Terasort up to 4.25x faster



Big Red II vs. Quarry for Terasort on Lustre

- Motivation & Design of HPCHadoop
- HPCHadoop demo
- Benchmarking Methodology
- Benchmark Results
- **Future Work**

Future Work

- Initial incarnation of HPCHadoop is a proof of concept, several additional features are in the works
- Support for schedulers other than PBS
- Further optimization for HiBench and comparison to other MapReduce systems like MARIANE
- Support for native use of shared parallel file systems such as GPFS and Lustre
  - Have begun collaborating with Intel on their Lustre compatibility module

Conclusions

- Hadoop is not for everyone and does not solve all "Big Data" problems
- However, for the problems that fit well into a Hadoop framework HPC resources are sometimes the only computational option for researchers
- HPCHadoop allows for easy set up and launching of Hadoop jobs on batch scheduled systems including the Cray X-series

Questions?

email questions to: **scamicha@iu.edu**
https://github.com/scamicha/HPCHadoop