MS Data Analysis with a touch of Spark

Animesh Sharma[1], Steven M. Lewis[2], Gurvinder S. Dahiya[3] and Geir Slupphaug[1,4]

1 The Proteomics and Metabolomics Core Facility (PROMEC), Norwegian University of Science and Technology (NTNU), Trondheim, Norway,
2 Institute for Systems Biology, Seattle, Washington, United States,
3 UNINETT, Trondheim, Norway,
4 Department of Cancer Research and Molecular Medicine, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Abstract

BACKGROUND-AIM: Modern protein analysis uses tandem mass spectrometry (MS) to generate spectra and searching all possible spectra for the best fit. The search space may involve thousands of millions of correlations thus an efficient algorithm is used to explore this combinatorial space. Such analyses strain the resources of existing machines and are limited in the size of problem that can be considered. Further adding more features to the search space, such as Post-Translational Modifications (PTMs) which are typical for proteins, scale exponentially with the number of PTMs. Thus a framework is needed which scales easily with the amount of data and search space..

METHODS: We have developed SparkHydra [https://github.com/gurvindersingh/sparkhydra], an Apache Spark [http://spark.apache.org/] based implementation of a widely used search algorithm COMET [http://cometms.sourceforge.net/] using data stored on a distributed file system HDFS. However Parquet database is used to store all possible fragments which optimizes the search. The algorithm has a dozen stages and has been used to score over 1 million spectra. Search is performed on a cluster and performance can scale with the number of machines applied to the problem. These spectra has been scored against whole reviewed Uniprot database [download July 2014] whose tryptic digest yields more than 2 billion peptides.

RESULTS: The search which takes roughly about 9 hours on a single machine can run within 3 hours on a spark cluster. The performance scales well with increasing the number of machines in Spark cluster and the top scoring hits correlate very well with the basic version of the COMET.

CONCLUSIONS: The framework gives researchers a scalable means to greatly expand analysis of proteins in their samples. Further it provides access to re-analyze the historical data from archives such as PRIDE in context of their interest. Along with providing a Spark REPL [https://spark.apache.org/docs/latest/quick-start.html] interface for interactive interpretation of the results, we also plan to extend it to utilize a popular interface SearchGUI [https://code.google.com/p/searchgui/] for setting up the search and PeptideShaker [https://code.google.com/p/peptide-shaker/] for presenting the search result.

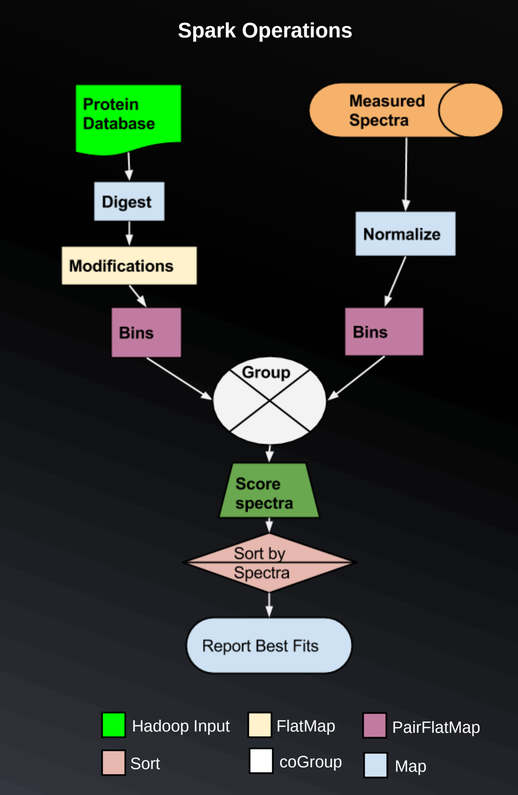
Proteomics as a big data challenge for cloud computing

Why?

- UniProtKB/TrEMBL has about 50 million protein sequences
- Typical MS run results in about 100,000 spectra
- Usually for quality control there is a ‘Decoy’ for every protein which result in increase by factor > 2
- Protein modifications and missed cleavages leads to exponential increase in number of peptides
- The search space is proportional to peptides * spectra
- Scoring peptides can be performed in parallel
- Task handled on a single machine can be mapped to cluster

Suitable for Parallel Execution

- Compute engine providers such as Amazon, Google offers scalable compute and storage infrastructure
- Cloud Computing allows parallel execution of tasks on large number of (virtual) machines
- As problem size increases more machines can be added
- But with many machines, what can fail, will fail sometime…
  - While data is moved
  - Memory is shared
  - CPU with different architecture



Apache Spark

- Provides reliable & distributed computation Framework
  1. Resilient data sets
  2. Control to distribute tasks nodes
  3. Control to manage failure
- Allows code to be written without knowledge of how parallelism will happen

Local Cluster management framework
https://in.uninett.no/category/projects/daas/
3 Master: 2 HDFS,1Mesos
9 Worker: 4core@1.87GHz, 20GB, 6TB 7200RPM SATA

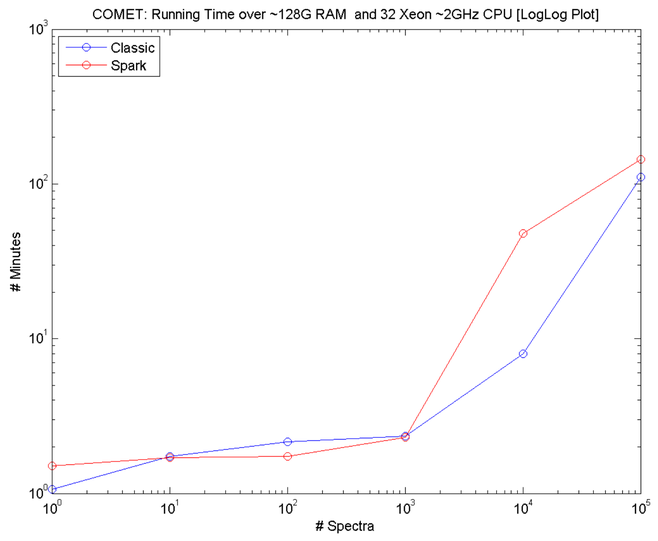
Resilient Distributed Datasets:
distributed
memory abstraction
fault-tolerance

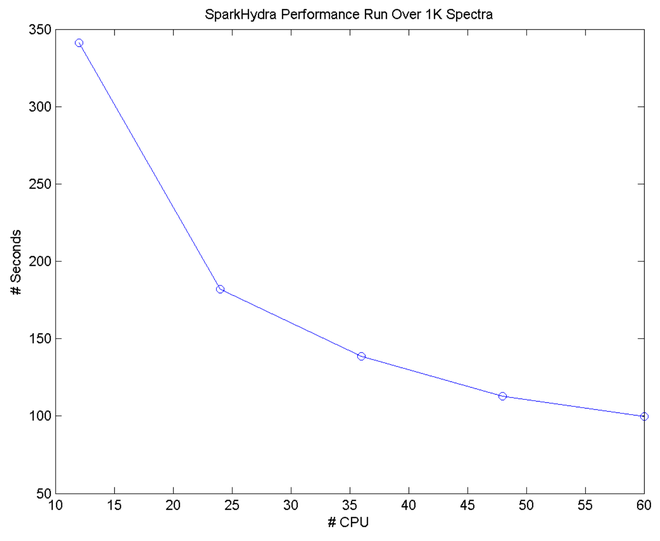
Functional beauty
lazy evaluation

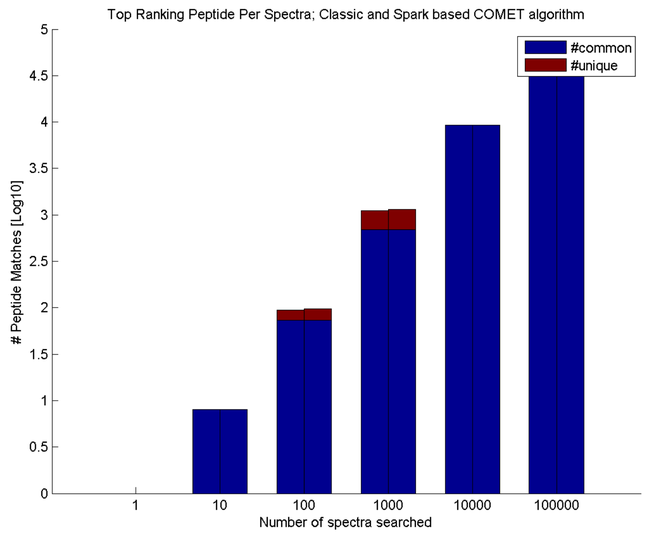
REPL:
Scala
Python
R

Peptide Fetch and Bin Pseudocode

```
// read proteins from hdfs - for large files this happens on many machines
RDD<Protein> proteins = HadooopRead(myFastaFile,FastaInputFormat.class);
// apply digester - flatMap returns 0..N results for a single input
RDD<Peptide> digest= proteins.flatMap(digestFunction);
// apply modifications- flatMap returns 0..N results for a single input
RDD<Peptide> withModifications = digest.flatMap(addModificationsFunction);
// apply modifications- flatMap returns 0..N results for a single input
PairRDD<Bin,Peptide> inBins= withModifications .mapToPair(binningFunction);
```







Advantages

- Moore's Law: Free lunch is over! A Fundamental Turn Toward Concurrency in Software – Herb Sutter [http://www.gotw.ca/publications/concurrency-ddj.htm]
  On Intel chips, we reached 2GHz a long time ago (August 2001)
  Physical limits; light isn’t getting any faster!
- Spark provides a scale-out alternative to classical scale-up approach
- Our experiments show Spark's reliability and scalability in Proteomics
- Spark uses distributed RAM, unlike hadoop, as well as disks to enhance performance
  “space is speed”
- Naturally extends to other Spark based Bioinformatics libraries
  Genomics: ADAM [http://bdgenomics.org/]
  Neuroinformatics: Thunder [http://thefreemanlab.com/thunder]
- Supports for large scale machine learning [MLlib: https://spark.apache.org/mllib/] and parallel computations over graphs [GraphX: https://spark.apache.org/graphx/] comes for free as APIs

Goals

1. Integrate our solution into existing pipeline [SearchGUI, PeptideShaker]
2. Support multiple input formats natively [currently MGF and mzXML]
3. Create a plugin like architecture for multiple search algorithms [currently COMET]
4. REPL to do interactive analysis
5. Provide a workflow like interface to use other Spark libraries [Mllib]
6. Ability to share workflow across researchers and (re)run analysis with different search parameters
   1. Enable historical MS data analysis
   2. Handle more modifications
   3. Search larger databases
   4. ADAM output as an input [search database]

Poster ID: 173