Improving LC-MS Data Analysis Pipelines to Leverage Distributed Compute Engines

Iman Mohtashemi*. Hugo Kitano, Andrew Nichols, Seth Just, Jian Wang, Theo Platt, and Serafim Batzoglou

Seer, Inc., Redwood City, CA 94065, USA

An automated, scalable proteomics data analysis workflow

Liquid chromatography coupled with mass spectrometry (LC-MS) has grown into a ubiquitous detection platform due to its speed, sensitivity, and applications. While instrumentation hardware continues to improve, the concurrent increase in translation from data to insight remains a bottleneck. Previously, we have demonstrated a cloud-based serverless task-based infrastructure where closed-source legacy algorithms are deployed as containerized applications leveraging AWS elastic container service. These algorithms are orchestrated with AWS services such as lambda functions and step functions. In this work, we focus on scaling label-free LC-MS data analysis workflows to enable large cohort studies using open-source algorithms leveraging distributed computing models in our AWS infrastructure.

Challenges

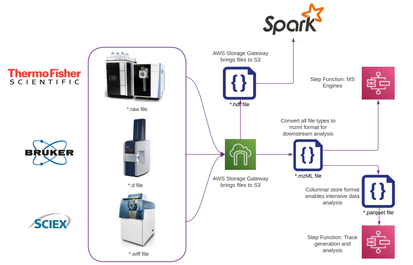
- Most LC-MS data analysis solutions are built for desktop environments and are closed-source 'black-box' executables and cannot be distributed natively
- Differential proteomics data analysis of large data sets ('group runs') require data aggregation which is memory/disk limited
- Existing applications are not designed for increasing compute and memory
- There is a need to modularize the ever-growing collection of applications for both DDA and DIA acquired LC-MS data

Solution

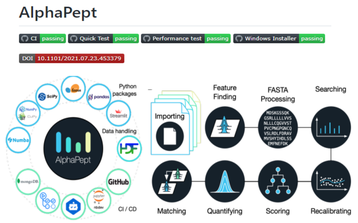
A carefully curated AWS proteomics data analysis workflow with choices, error handling, and exception fallbacks including:

- **Automated file transfer** to the cloud and **conversion** to standard **mzML**, **parquet** and **HDF5** filetypes
- **Automate single file analysis** for every injection upon raw data file arrival
- User-specified **group run analyses** with pre-defined recipes and settings (possible with 1000s of files)
- **Spark-accelerated modular workflows** built on top of open-source Alphapept

LC-MS Cloud Connectivity

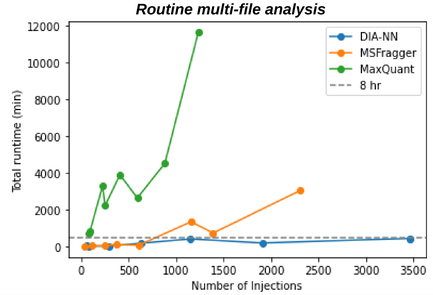


Open-Source Code Base



https://github.com/MannLabs/alphapept

Routine multi-file analysis

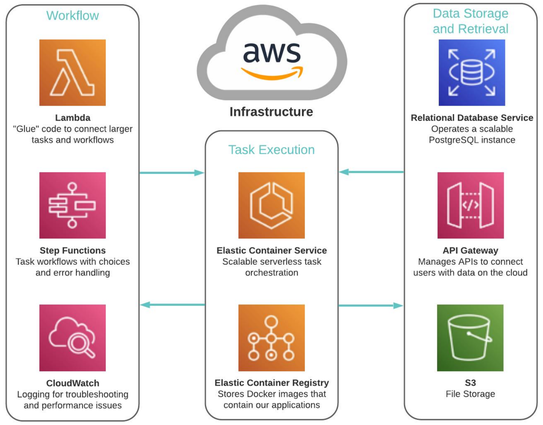


- **Most processing is embarrassingly parallel and scales well**
- **Key steps (*e.g.*, alignment, quant) require many more resources, increasing cost and limiting run size**
- **DIA-NN can process 1000s of samples in under 8 hours (without MBR)**
- **Scaling beyond ~5000 samples will require:**
  - Modularization of pipelines
  - Efficient data access at scale
  - Distributed implementations of key algorithms

Proteomics Pipelines with a smart cloud infrastructure

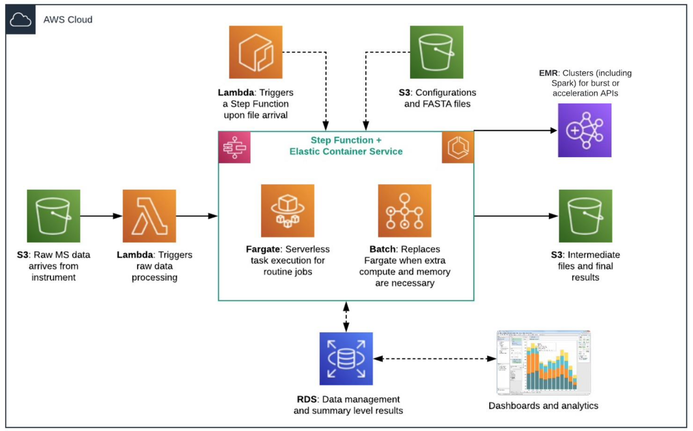
A combination of AWS services to process, store, and retrieve LC-MS data

The AWS ecosystem at Seer



Multiple cloud services working in harmony

The coordination of automated file analysis from MS instruments to data storage with cluster computing APIs



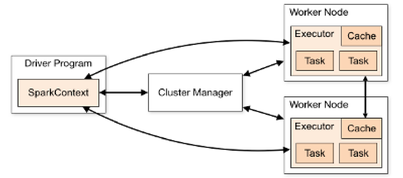
Application of open-source modular infrastructures; Alphapept and Apache Spark

To improve scalability limits and large-scale proteomics data analysis infrastructure we have evaluated the recently published Alphapept platform[1]. Here are highlighted benefits of this platform:

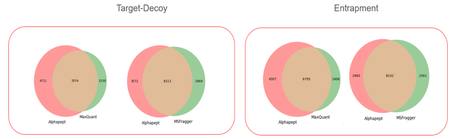
- Python programming language, has easy-to-understand syntax with excellent support of scientific libraries making it easier for developers from different backgrounds to contribute to and implement new ideas. Furthermore, all major cloud vendors support the python language for distributed computing (e.g., pyspark)
- Easy on-ramp for community validation and contributions through the concept of literate programming, implemented in Jupyter Notebooks of the different modules. A baseline framework for continuous integration, testing, and benchmarking enforces solid software engineering principles
- Efficient HDF5 file formatting and just-in-time machine code compilation on CPU and GPU, achieving hundred-fold speed improvements while maintaining clear syntax and rapid development speed
- Distributed computing potential using AWS elastic map reduce (EMR) and Pyspark

Integrating Alphapept with Apache Spark

- Alignment is an $O(n^2)$ operation, requiring nC2 operations comparing each of the n files to each other
- Data from each of the n files should be read once, before any calculations, to eliminate any redundant I/O operations
- When each parallelized task needs access to read-only file data, Spark supplies "broadcasting" to speed up the transfer of data into each executor.
- Broadcasting caches data into every executor, enabling each execution to begin almost instantaneously



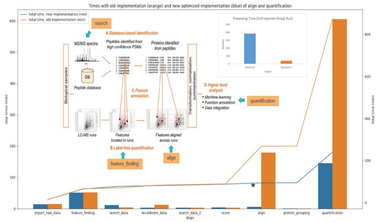
Search Engine Comparisons



Engine	Unique Peptides	False Match Rate (FMR)
MaxQuant	8201	0
MSFragger	10703	0.01
Alphapept	11102	0.04

Using target/decoy and entrapment analysis we demonstrate Alphapept's search strategy in comparison with other search engines at a reasonable FMR.

Label-free Quan Pipeline



We integrate pyspark for relieving computational bottlenecks where data aggregation is required such as chromatographic alignment.

Results

A next-generation platform capable of analyzing large cohort proteomics studies in hours supporting fleets of vendor neutral LCMS instruments

- Supporting hundreds of terabytes of incoming LCMS data annually
- Enabling large cohort group runs
- Spark-accelerated workflows supporting thousands of group run analysis

References

[1]Strauss et al., 2021 BioRxiv

Publications

Seer, Inc., Redwood City, CA – hkitano@seer.bio