NSF14-43054 started October 1, 2014

Datanet: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science

- Indiana University (Fox, Qiu, Crandall, von Laszewski)
- Rutgers (Jha)
- Virginia Tech (Marathe)
- Kansas (Paden)
- Stony Brook (Wang)
- Arizona State (Beckstein)
- Utah (Cheatham)

Overview by Geoffrey Fox, May 16, 2016

http://spidal.org/
Some Important Components of SPIDAL Dibbs

- **NIST Big Data Application Analysis** – features of data intensive Applications.
- **HPC-ABDS**: Cloud-HPC interoperable software **performance of HPC** (High Performance Computing) and the rich **functionality** of the commodity **Apache Big Data Stack**.
  - This is a reservoir of software subsystems – nearly all from outside the project and being a mix of HPC and Big Data communities.
  - Leads to **Big Data – Simulation – HPC Convergence**.
- **MIDAS**: Integrating Middleware – from project.
- **Applications**: Biomolecular Simulations, Network and Computational Social Science, Epidemiology, Computer Vision, Spatial Geographical Information Systems, Remote Sensing for Polar Science and Pathology Informatics.
- **SPIDAL (Scalable Parallel Interoperable Data Analytics Library)**: Scalable Analytics for:
  - Domain specific data analytics libraries – mainly from project.
  - Add Core Machine learning libraries – mainly from community.
  - Performance of Java and MIDAS Inter- and Intra-node.
- **Benchmarks** – project adds to community WBDB2015 Benchmarking Workshop
- **Implementations**: XSEDE and Blue Waters as well as clouds (OpenStack, Docker)
Big Data - Big Simulation (Exascale) Convergence

• Discuss **Data** and **Model** together as built around problems which combine them, but we can get insight by separating which allows better understanding of **Big Data - Big Simulation “convergence”**

• Big Data implies Data is large but Model varies
  – e.g. **LDA** with many topics or **deep learning** has large model
  – Clustering or Dimension reduction can be quite small

• **Simulations** can also be considered as **Data** and **Model**
  – **Model** is solving particle dynamics or partial differential equations
  – **Data** could be small when just boundary conditions
  – **Data** large with data assimilation (weather forecasting) or when data visualizations are produced by simulation

• **Data** often static between iterations (unless streaming); **Model** varies between iterations

• Take 51 NIST and other use cases \(\rightarrow\) derive multiple specific features

• Generalize and systematize with features termed “facets”

• **50 Facets (Big Data) or 64 Facets (Big Simulation and Data)** divided into 4 sets or views where each view has “similar” facets
  – Allows one to study coverage of benchmark sets and architectures
64 Features in 4 views for Unified Classification of Big Data and Simulation Applications

**Simulations**
- Nature of mesh if used
- Evolution of Discrete Systems
- N-body Methods
- Spectral Methods
- Multiscale Method
- Iterative PDE Solvers
- Data Alignment
- Streaming Data Algorithms
- Optimization Methodology
- Learning
- Data Classification/Query/index
- Recommender Engine
- Base Data Statistics
- Core Libraries
- Graph Algorithms
- Linear Algebra Kernels/Many subclasses
- Global (Analytics/Informatics/Simulations)
- Local (Analytics/Informatics/simulations)
- Micro-benchmarks

**Analytics (Model for Data)**
- Big Data Processing
- Diamonds: Simulations Analytics (Model for Data)
- Nearly all Data+Model
- Both

**Processing View (All Model)**
- Geospatial Information System
- HPC Simulations
- Internet of Things
- Metadata/Provenance
- Shared / Dedicated / Transient / Permanent
- Archived/Batched/Streaming – S1, S2, S3, S4, S5
- HDFS/Lustre/GPFS
- Files/Objects
- Enterprise Data Model
- SQL/NoSQL/NewSQL

**Problem Architecture View (Nearly all Data+Model)**

**Convergence Diamonds Views and Facets**
- Performance Metrics
- Execution Environment; Core libraries
- Flops per Byte/Byte/Memory/IO/Flops per watt
- Flops
- Data Volume
- Model Size
- Data Velocity
- Data Variety
- Model Variety
- Veracity
- Communication Structure
- Dynamic = D / Static = S
- Regular = R / Irregular = I Model
- Iterative / Simple
- Nature of mesh if used
- Consistency

**Execution View (Mix of Data and Model)**
- Regular
- Model Abstraction
- Model Metric
- Non-Metric
- Nature of mesh if used
6 Forms of MapReduce

Cover “all” circumstances

Describes
- Problem (Model reflecting data)
- Machine
- Software

Architecture
# HPC-ABDS

## Kaleidoscope of (Apache) Big Data Stack (ABDS) and HPC Technologies

HPC-ABDS Mapping of Activities

Green is MIDAS

- **Level 17: Orchestration**: Apache Beam (Google Cloud Dataflow) integrated with Cloudmesh on HPC cluster
- **Level 16: Applications**: Datamining for molecular dynamics, Image processing for remote sensing and pathology, graphs, streaming, bioinformatics, social media, financial informatics, text mining
- **Level 16: Algorithms**: Generic and custom for applications **SPIDAL**
- **Level 14: Programming**: Storm, Heron (Twitter replaces Storm), Hadoop, Spark, Flink. Improve Inter- and Intra-node performance
- **Level 13: Communication**: Enhanced Storm and Hadoop using HPC runtime technologies, Harp
- **Level 11: Data management**: Hbase and MongoDB integrated via use of Beam and other Apache tools; enhance Hbase
- **Level 9: Cluster Management**: Integrate Pilot Jobs with Yarn, Mesos, Spark, Hadoop; integrate Storm and Heron with Slurm
- **Level 6: DevOps**: Python Cloudmesh virtual Cluster Interoperability
Big Data ABDS

17. Orchestration  Beam, Crunch, Tez, Cloud Dataflow
16. Libraries  MLlib/Mahout, TensorFlow, CNTK, R, Python
15A. High Level Programming  Pig, Hive, Drill
15B. Platform as a Service  App Engine, BlueMix, Elastic Beanstalk
Languages  Java, Erlang, Scala, Clojure, SQL, SPARQL, Python
14B. Streaming  Storm, Kafka, Kinesis
13,14A. Parallel Runtime  Hadoop, MapReduce
2. Coordination  Zookeeper
12. Caching  Memcached
11. Data Management  Hbase, Accumulo, Neo4J, MySQL
10. Data Transfer  Sqoop
9. Scheduling  Yarn, Mesos
8. File Systems  HDFS, Object Stores
1, 11A Formats  Thrift, Protobuf
5. IaaS  OpenStack, Docker
Infrastructure  CLOUDS

HPC-ABDS Integrated Software

HPC, Cluster
Kepler, Pegasus, Taverna
ScaLAPACK, PETSc, Matlab
Domain-specific Languages
XSEDE Software Stack
Fortran, C/C++, Python
MPI/OpenMP/OpenCL
CUDA, Exascale Runtime
iRODS
GridFTP
Slurm
Lustre
FITS, HDF
Linux, Bare-metal, SR-IOV
SUPERCOMPUTERS
MIDAS: Software Activities in DIBBS

• Developing **HPC-ABDS** concept to integrate HPC and Apache Technologies
• **Java**: relook at Java Grande to make performance “best simply possible”
• **DevOps**: Cloudmesh provides interoperability between HPC and Cloud (OpenStack, AWS, Docker) platforms based on virtual clusters with software defined systems using Ansible (Chef)
• **Scheduling**: Integrate Slurm and Pilot jobs with Yarn & Mesos (ABDS schedulers), Programming layer (Hadoop, Spark, Flink, Heron/Storm)
• **Communication** and scientific **data abstractions**: Harp plug-in to Hadoop outperforms ABDS programming layers
• **Data Management**: use Hbase, MongoDB with customization
• **Workflow**: Use Apache Crunch and Beam (Google Cloud Data flow) as they link to other ABDS technologies.
• Starting to integrate MIDAS components and move into algorithms of SPIDAL Library
Java MPI performs better than Threads
128 24 core Haswell nodes on SPIDAL DA-MDS Code

SM = Optimized
Shared memory for intra-node MPI

Best MPI; inter and intra node

Best Threads intra node
And MPI inter node

HPC into Java Runtime and Programming Model
Cloudmesh Interoperability DevOps Tool

• **Model:** Define software configuration with tools like Ansible; instantiate on a virtual cluster
• An easy-to-use command line program/shell and portal to interface with heterogeneous infrastructures
  – Supports OpenStack, AWS, Azure, SDSC Comet, virtualbox, libcloud supported clouds as well as classic HPC and Docker infrastructures
  – Has an abstraction layer that makes it possible to integrate other IaaS frameworks
  – Uses defaults that help interacting with various clouds
  – Managing VMs across different IaaS providers is easy
  – The client *saves state* between consecutive calls
• Demonstrated interaction with various cloud providers:
  – FutureSystems, Chameleon Cloud, Jetstream, CloudLab, Cybera, AWS, Azure, virtualbox
• **Status:** AWS, and Azure, VirtualBox, Docker need improvements; we focus currently on Comet and NSF resources that use OpenStack
• Currently evaluating 40 team projects from “Big Data Open Source Software Projects Class” which used this approach running on VirtualBox, Chameleon and FutureSystems
Cloudmesh Interoperability DevOps Tool

- **Model:** Define software configuration with tools like Ansible; instantiate on a virtual cluster
- An easy-to-use command line program/shell and portal to interface with heterogeneous infrastructures
  - Supports OpenStack, AWS, Azure, SDSC Comet, virtualbox, libcloud supported clouds as well as classic HPC and Docker infrastructures
  - Has an abstraction layer that makes it possible to integrate other IaaS frameworks
  - Uses defaults that help interacting with various clouds
  - Managing VMs across different IaaS providers is easy
  - The client *saves state* between consecutive calls
- Demonstrated interaction with various cloud providers:
  - FutureSystems, Chameleon Cloud, Jetstream, CloudLab, Cybera, AWS, Azure, virtualbox
- **Status:** AWS, and Azure, VirtualBox, Docker need improvements; we focus currently on Comet and NSF resources that use OpenStack
- Currently evaluating 40 team projects from “Big Data Open Source Software Projects Class” which used this approach running on VirtualBox, Chameleon and FutureSystems
Cloudmesh Client - Architecture

Component View

Layered View

Cloudmesh Client

Choreography
- Paas Launchers
- Workflow
- Group Management
- Reservation

Access
- Command Shell & Line
- REST
- Access Abstraction

Systems Configuration
- Database Abstraction
  - create, read, update, delete (CRUD)
- SQLite
- MongoDB

Security
- Policy Management
- Authorization
- Authentication

Compute
- Virtual Cluster Abstraction
- create, read, update, delete
- Batch Queue Abstraction
- VM Abstraction
- Container Abstraction

Cloudmesh Access

Cloudmesh Shell

Commandline

Indiana University Bloomington
School of Informatics and Computing
Cloudmesh Client – OSG management

OSG: Open Science Grid. LIGO data analysis was conducted on Comet supported by the Cloudmesh client. Funded by NSF Comet.

- While using Comet it is possible to use the same images that are used on the internal OSG cluster.
- This reduces overall management effort.
- The client is used to manage the VMs.
Cloudmesh Client – In support of Experiment Workflow

- Manage VMs and virtual clusters
- Scripts, Variables, Shared state
- Integration into Ipython and Apache Beam
- Cloudmesh scripts, Integrate with shell
- Integrate with Ansible
- Buildin gcop and rsync commands
- Choose general Infrastructure HPC to Clouds

Cloudmesh Client

Create IaaS

Deploy PaaS

Deploy Data

Evaluate

Execute Scripts

Repeat
Pilot-Hadoop/Spark Architecture

HPC into Scheduling Layer

<table>
<thead>
<tr>
<th>Application</th>
<th>Application-level Scheduling</th>
<th>System-level Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Reduce</td>
<td>YARN</td>
<td>HPC Scheduler (Slurm, Torque, SGE)</td>
</tr>
<tr>
<td>Spark-App</td>
<td>Spark</td>
<td>YARN/HDFS</td>
</tr>
<tr>
<td>Other YARN App</td>
<td>Hadoop/Spark App (e.g. Spark, Tez, LLama)</td>
<td></td>
</tr>
<tr>
<td>Hadoop/Spark App</td>
<td>HPC App (e.g. MPI)</td>
<td>Pilot-Job</td>
</tr>
</tbody>
</table>

Mode I: Hadoop on HPC

Mode II: HPC on Hadoop

Pilot-Hadoop Example
Pilot-Data/Memory for Iterative Processing

Pilot-Manager
- Compute-Data Manager
- Pilot-Compute Manager
- Pilot-Data Manager

Distributed Coordination Service

Pilot-Agent
- Compute Manager
  - SSH
  - MPI
  - YARN
  - Spark
- Distributed Memory Manager
  - File
  - Redis
  - Spark
- Data Manager
  - SSH
  - iRODS
  - HDFS
  - Cloud

Resource Slot
- Compute-Unit
- In-Memory Data-Unit Chunk
- Data-Unit Chunk

Scalable K-Means

Provide common API for distributed cluster memory
Harp Implementations

- Basic Harp: Iterative communication and scientific data abstractions
- Careful support of distributed data AND distributed model
- Avoids parameter server approach but distributes model over worker nodes and supports collective communication to bring global model to each node
- Applied first to Latent Dirichlet Allocation LDA with large model and data
Latent Dirichlet Allocation on 100 Haswell nodes: red is Harp (lgs and rtt)

Clueweb

- Model Likelihood vs Execution Time (s)
  - lgs
  - Yahoo!LDA
  - lgs-4s

Bi-gram

- Model Likelihood vs Execution Time (s)
  - rtt
  - Petuum

enwiki

- Model Likelihood vs Execution Time (s)
  - lgs
  - Yahoo!LDA
Harp LDA Scaling Tests

Harp LDA on Big Red II Supercomputer (Cray)

- **Big Red II**: tested on 25, 50, 75, 100 and 125 nodes; each node uses 32 parallel threads; Gemini interconnect

Harp LDA on Juliet (Intel Haswell)

- **Juliet**: tested on 10, 15, 20, 25, 30 nodes; each node uses 64 parallel threads on 36 core Intel Haswell nodes (each with 2 chips); Infiniband interconnect

Corpus: 3,775,554 Wikipedia documents, Vocabulary: 1 million words; Topics: 10k topics; alpha: 0.01; beta: 0.01; iteration: 200
SPIDAL Algorithms – Subgraph mining

• Finding patterns in graphs is very important
  – Counting the number of embeddings of a given labeled/unlabeled template subgraph
  – Finding the most frequent subgraphs/motifs efficiently from a given set of candidate templates
  – Computing the graphlet frequency distribution.

• Reworking existing parallel VT algorithm Sahad with MIDAS middleware giving HarpSahad which runs 5 (Google) to 9 (Miami) times faster than original Hadoop version

• Work in progress
SPIDAL Algorithms – Random Graph Generation

- Random graphs, important and needed with particular degree distribution and clustering coefficients.
  - Preferential attachment (PA) model, Chung-Lu (CL), stochastic Kronecker, stochastic block model (SBM), and block two-level Erdos-Renyi (BTER)
  - Generative algorithms for these models are mostly sequential and take a prohibitively long time to generate large-scale graphs.
- SPIDAL working on developing efficient parallel algorithms for generating random graphs using different models with new DG method with low memory and high performance, almost optimal load balancing and excellent scaling.
  - Algorithms are about 3-4 times faster than the previous ones.
  - Generate a network with 250 billion edges in 12 seconds using 1024 processors.
- Needs to be packaged for SPIDAL using MIDAS (currently MPI)
SPIDAL Algorithms – Triangle Counting

- Triangle counting; important special case of subgraph mining and specialized programs can outperform general program
- Previous work used Hadoop but MPI based PATRIC is much faster
- SPIDAL version uses much more efficient decomposition (non-overlapping graph decomposition) – a factor of 25 lower memory than PATRIC
- Next graph problem – Community detection

MPI version complete. Need to package for SPIDAL and add MIDAS -- Harp
SPIDAL Algorithms – Core I

• Several parallel core machine learning algorithms; need to add SPIDAL Java optimizations to complete parallel codes except MPI MDS

• \(O(N^2)\) distance matrices calculation with Hadoop parallelism and various options (storage MongoDB vs. distributed files), normalization, packing to save memory usage, exploiting symmetry

• WDA-SMACOF: Multidimensional scaling MDS is optimal nonlinear dimension reduction enhanced by SMACOF, deterministic annealing and Conjugate gradient for non-uniform weights. Used in many applications
  – MPI (shared memory) and MIDAS (Harp) versions

• MDS Alignment to optimally align related point sets, as in MDS time series

• WebPlotViz data management (MongoDB) and browser visualization for 3D point sets including time series. Available as source or SaaS

• MDS as \(\chi^2\) using Manxcat. Alternative more general but less reliable solution of MDS. Latest version of WDA-SMACOF usually preferable

• Other Dimension Reduction: SVD, PCA, GTM to do
SPIDAL Algorithms – Core II

• **Latent Dirichlet Allocation LDA** for topic finding in text collections; new algorithm with MIDAS runtime outperforming current best practice

• **DA-PWC Deterministic Annealing Pairwise Clustering** for case where points aren’t in a vector space; used extensively to cluster DNA and proteomic sequences; improved algorithm over other published. Parallelism good but needs SPIDAL Java

• **DAVS Deterministic Annealing Clustering for vectors**; includes specification of errors and limit on cluster sizes. Gives very accurate answers for cases where distinct clustering exists. Being upgraded for new LC-MS proteomics data with one million clusters in 27 million size data set

• **K-means basic vector clustering**: fast and adequate where clusters aren’t needed accurately

• **Elkan’s improved K-means vector clustering**: for high dimensional spaces; uses triangle inequality to avoid expensive distance calcs

• **Future work – Classification**: logistic regression, Random Forest, SVM, (deep learning); Collaborative Filtering, TF-IDF search and Spark **MLlib** algorithms

• **Harp-DaaL** extends Intel DAAL’s local batch mode to multi-node distributed modes
  – Leveraging Harp’s benefits of communication for iterative compute models
SPIDAL Algorithms – Optimization I

- **Manxcat**: Levenberg Marquardt Algorithm for non-linear $\chi^2$ optimization with sophisticated version of Newton’s method calculating value and derivatives of objective function. Parallelism in calculation of objective function and in parameters to be determined. **Complete – needs SPIDAL Java optimization**

- **Viterbi** algorithm, for finding the maximum a posteriori (MAP) solution for a Hidden Markov Model (HMM). The running time is $O(n*s^2)$ where $n$ is the number of variables and $s$ is the number of possible states each variable can take. We will provide an "embarrassingly parallel" version that processes multiple problems (e.g. many images) independently; parallelizing within the same problem not needed in our application space. **Needs Packaging in SPIDAL**

- **Forward-backward algorithm**, for computing marginal distributions over HMM variables. Similar characteristics as Viterbi above. **Needs Packaging in SPIDAL**
Loopy belief propagation (LBP) for approximately finding the maximum a posteriori (MAP) solution for a Markov Random Field (MRF). Here the running time is $O(n^2s^2i)$ in the worst case where $n$ is number of variables, $s$ is number of states per variable, and $i$ is number of iterations required (which is usually a function of $n$, e.g. $\log(n)$ or $\sqrt{n}$). Here there are various parallelization strategies depending on values of $s$ and $n$ for any given problem.

- We will provide two parallel versions: embarrassingly parallel version for when $s$ and $n$ are relatively modest, and parallelizing each iteration of the same problem for common situation when $s$ and $n$ are quite large so that each iteration takes a long time relative to number of iterations required.

- Needs Packaging in SPIDAL

Markov Chain Monte Carlo (MCMC) for approximately computing marking distributions and sampling over MRF variables. Similar to LBP with the same two parallelization strategies. Needs Packaging in SPIDAL
Imaging Applications: Remote Sensing, Pathology, Spatial Systems

• Both Pathology/Remote sensing working on 3D images
• Each pathology image could have 10 billion pixels, and we may extract a million spatial objects per image and 100 million features (dozens to 100 features per object) per image. We often tile the image into 4K x 4K tiles for processing. We develop buffering-based tiling to handle boundary-crossing objects. For each typical research study, we may have hundreds to thousands of pathology images.
• Remote sensing aimed at radar images of ice and snow sheets
• 2D problems need modest parallelism “intra-image” but often need parallelism over images
• 3D problems need parallelism for an individual image
• Use Optimization algorithms to support applications (e.g. Markov Chain, Integer Programming, Bayesian Maximum a posteriori, variational level set, Euler-Lagrange Equation)
• Classification (deep learning convolution neural network, SVM, random forest, etc.) will be important
2D Radar Polar Remote Sensing

- Need to estimate structure of earth (ice, snow, rock) from radar signals from plane in 2 or 3 dimensions.
- Original 2D analysis ([11]) used Hidden Markov Methods; better results using MCMC (our solution)

Extending to snow radar layers
3D Radar Polar Remote Sensing

- Uses LBP to analyze 3D radar images

Radar gives a cross-section view, parameterized by angle and range, of the ice structure, which yields a set of 2-d tomographic slices (right) along the flight path.

Each image represents a 3d depth map, with along track and cross track dimensions on the x-axis and y-axis respectively, and depth coded as colors.

Reconstructing bedrock in 3D, for (left) ground truth, (center) existing algorithm based on maximum likelihood estimators, and (right) our technique based on a Markov Random Field formulation.
Algorithms – Nuclei Segmentation for Pathology Images

- Segment boundaries of nuclei from pathology images and extract features for each nucleus
- Consist of tiling, segmentation, vectorization, boundary object aggregation
- Could be executed on MapReduce (MIDAS Harp)

Execution pipeline on MapReduce (MIDAS Harp)
**Algorithms – Spatial Querying Methods**

- **Hadoop-GIS** is a general framework to support high performance spatial queries and analytics for spatial big data on MapReduce.
- It supports multiple types of spatial queries on MapReduce through spatial partitioning, customizable spatial query engine and on-demand indexing.
- **SparkGIS** is a variation of Hadoop-GIS which runs on Spark to take advantage of in-memory processing.
- Will extend Hadoop/Spark to Harp MIDAS runtime.
- **2D complete; 3D in progress**

![Spatial Queries Diagram]

![Architecture of Spatial Query Engine Diagram]
Some Applications Enabled

- KU/IU: Remote Sensing in Polar Regions
- SB: Digital Pathology Imaging
- SB: Large scale GIS applications, including public health
- VT: Graph Analysis in studies of networks in many areas
- UT, ASU, Rutgers: Analysis of Biomolecular simulations

Applications not part of Dibbs project but algorithms/software used
- IU: Bioinformatics and Financial Modeling with MDS
- Integration with Network Science Infrastructure
  - VT/IU: CINET: SPIDAL algorithms will be made available
  - IU: Osome Observatory on Social Media, currently Twitter [https://peerj.com/preprints/2008/](https://peerj.com/preprints/2008/) using enhanced HBase
  - IU: Topic Analysis of text data
- IU/Rutgers: Streaming with HPC enhanced Storm/Heron
Enabled Applications – Digital Pathology

- Digital pathology images scanned from human tissue specimens provide rich information about morphological and functional characteristics of biological systems.
- Pathology image analysis has high potential to provide diagnostic assistance, identify therapeutic targets, and predict patient outcomes and therapeutic responses.
- It relies on both pathology image analysis algorithms and spatial querying methods.
- Extremely large image scale.
Applications – Public Health

- GIS-oriented public health research has a strong focus on the locations of patients and the agents of disease, and studies the spatial patterns and variations.
- Integrating multiple spatial big data sources at fine spatial resolutions allow public health researchers and health officials to adequately identify, analyze, and monitor health problems at the community level.
- This will rely on high performance spatial querying methods on data integration.
- Note synergy between GIS and Large image processing as in pathology.
Biomolecular Simulation Data Analysis

- Utah (CPPTraj), Arizona State (MDAnalysis), Rutgers
- Parallelize key algorithms including $O(N^2)$ distance computations between trajectories
- Integrate SPIDAL $O(N^2)$ distance and clustering libraries

Path Similarity Analysis (PSA) with Hausdorff distance

$$\delta_H(P, Q) = \max \left\{ \max_{p \in P} \min_{q \in Q} d(p, q), \max_{q \in Q} \min_{p \in P} d(q, p) \right\}$$
Clustered distances for two methods for sampling macromolecular transitions (200 trajectories each) showing that both methods produce distinctly different pathways.

RADICAL Pilot benchmark run for three different test sets of trajectories, using 12x12 “blocks” per task.
Classification of lipids in membranes

Biological membranes are lipid bilayers with distinct inner and outer surfaces that are formed by lipid mono layers (leaflets). Movement of lipids between leaflets or change of topology (merging of leaflets during fusion events) is difficult to detect in simulations.

Lipids colored by leaflet
Same color: continuous leaflet.
LeafletFinder

LeafletFinder is a graph-based algorithm to detect continuous lipid membrane leaflets in a MD simulation*. The current implementation is slow and does not work well for large systems (>100,000 lipids).

Phosphate atom coordinates
Build nearest-neighbors adjacency matrix
Find largest connected subgraphs

Time series of Stock Values projected to 3D
Using one day stock values measured from January 2004 and starting after one year
January 2005 (Filled circles are final values)