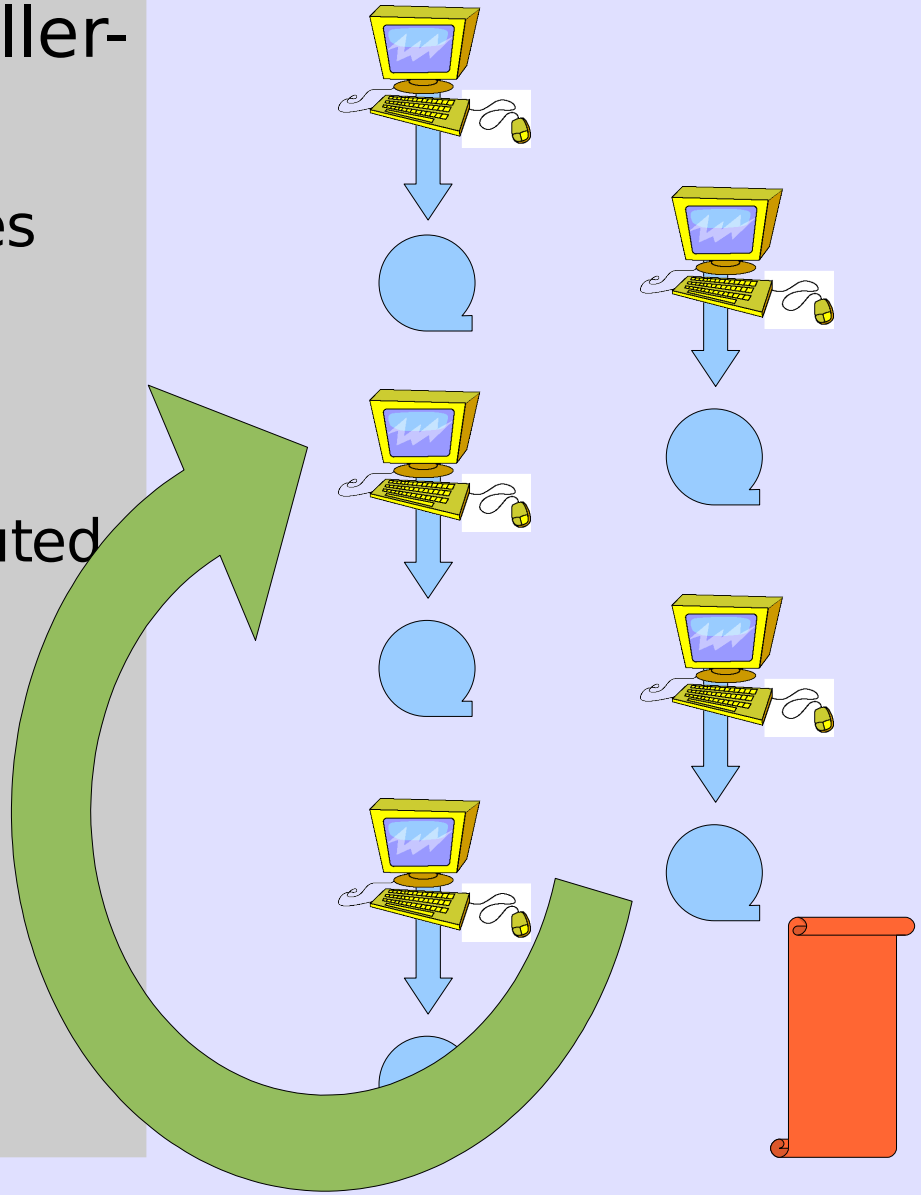
Mining the Grid Using the Grid

Work by:
Ran Wolff (U. Haifa, Israel)
Assaf Schuster (Technion)
Noam Palatin (Technion)
Arie Leizarowitz (Technion)

Supported by DataMiningGrid (FP6)
www.dataminingrid.org

Preface – Data Mining Grid

- Key idea – data mining is killer-app for grids
  - Batch process, many resources
  - Access mode for data-grids
- Technion demonstrator
  - Monitor the grid using distributed data mining on the grid
  - Several nice applications:
    - Misconfiguration detection
    - Job scheduling and meta-scheduling
    - Data pre-fetching



Mining the Grid for Misconfiguration Detection

- Data and knowledge
- **Architecture**
- Background knowledge
- **Algorithm**
- Implementation
- Validation

Data and Knowledge

Data

- Sources:
  - Logs (execution, etc.)
  - Configuration
  - Machine status (top, etc.)
- Collected on-line

host	RT SEC	Platform	condor Ver	VM MB	Disk MB	condor Load	Load	Mem MB	Arch	KFlops	Mips
bl2-05:vm1	537	ppc-linux	6.7.7	524	13166	0.706	1.529	1961	ppc64	1056216	3066
bl2-05:vm2	404	ppc-linux	6.7.7	524	13166	0.760	1.480	1961	ppc64	1056739	3083
bl2-06:vm1	486	ppc-linux	6.7.7	524	8850	1.403	2.351	1961	ppc64	1065326	3061
bl2-06:vm2	387	ppc-linux	6.7.7	524	8850	1.205	1.681	1961	ppc64	1065326	3061
bl2-07:vm1	515	ppc-linux	6.7.7	524	8379	1.331	1.997	1961	ppc64	1040999	3162
bl2-07:vm2	544	ppc-linux	6.7.7	524	8379	1.353	2.150	1961	ppc64	1040999	3162
bl2-08:vm1	555	ppc-linux	6.7.7	524	8464	1.338	2.216	1961	ppc64	1074788	3162
bl2-08:vm2	574	ppc-linux	6.7.7	524	8464	1.426	2.327	1961	ppc64	1074788	3162

Knowledge

- List of machines with abnormal behavior
- Short intelligible explanation for the abnormality
- Outliers:
  - “data points which deviate markedly from other ...”
  - Indicator: Average distance to k nearest neighbors
- Java jobs assigned to machine ng-12 terminate much faster then they would on similar machines

Architecture, Principles

- Zero Intrusion
  - No experienced overhead
  - No code changes
  - No special monitoring mode
- Discover unknown faults
  - knowledge discovery vs. pattern recognition
  - As much data as possible
- Intelligible output
  - Transform raw data using grid ontology

- Pool: 1000 machines
- Data rate: 1K / sec

- 86.4G / day
- 99.9% never used
  - Typical in KD
- Decentralized storage
  - No storage overhead
  - Comm. overhead proportional to output

Architecture, Choices

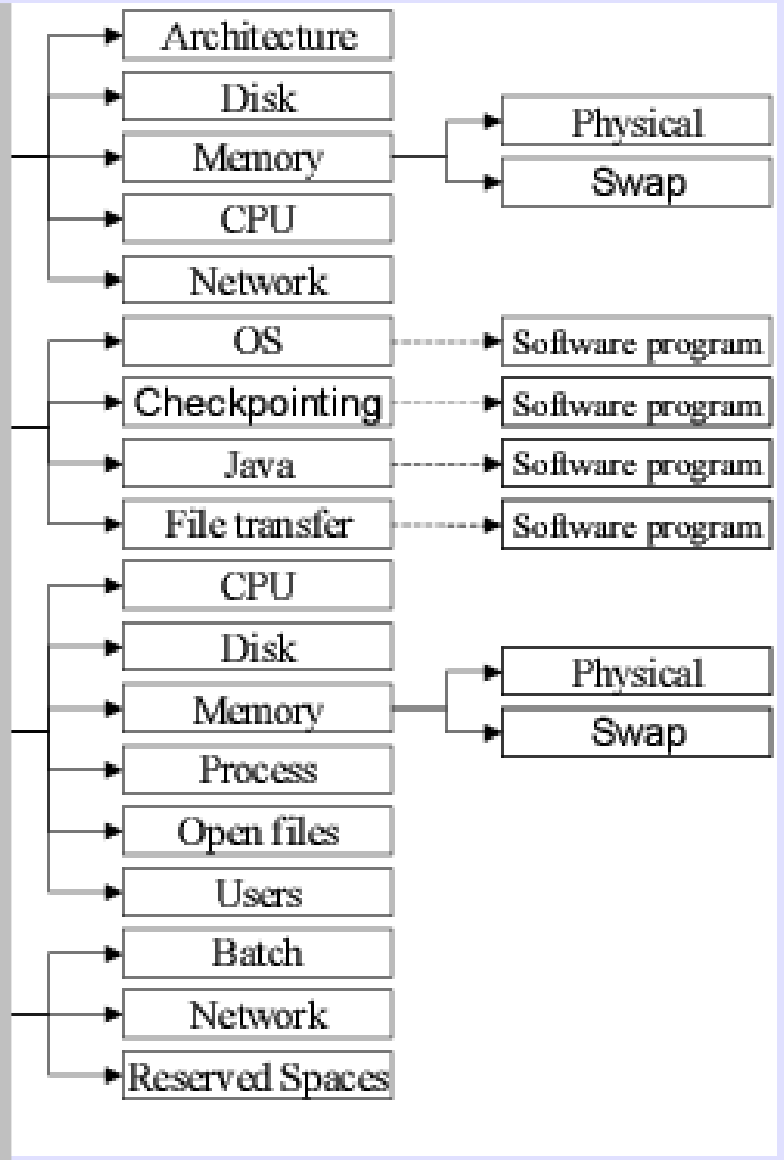
- Decentralized storage
  - Data is not moved unless required
- Decentralized processing
  - Distributed KD algorithm
  - Limited availability
  - No synchronization & termination

Layered System

- Acquisition (local):
  - External to grid software
  - System specific
- Presentation (local):
  - Reported to grid software
  - Ontology dependent
- Discovery (global):
  - A grid application
  - System independent

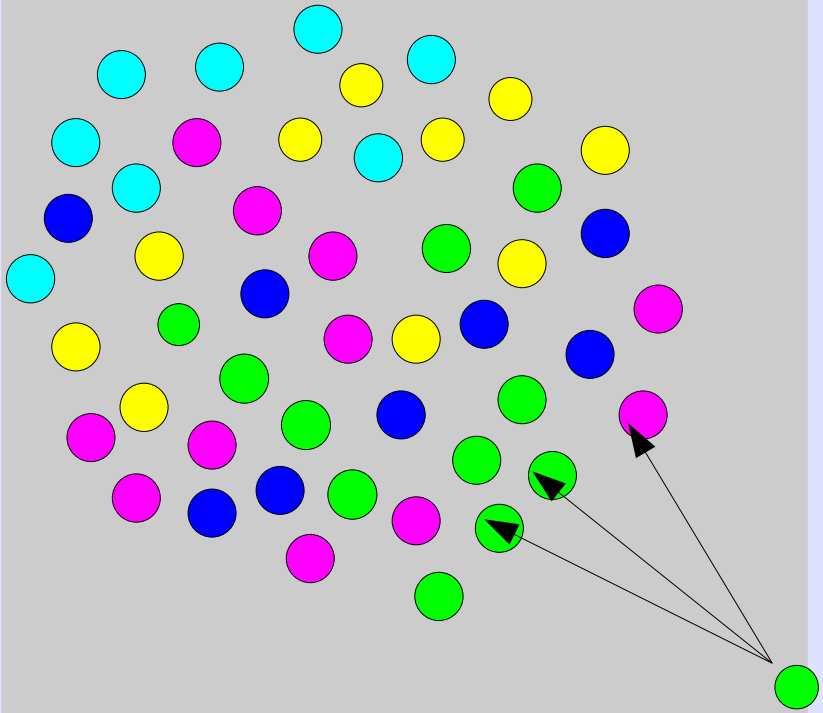
Background Knowledge

- Background knowledge is essential for
  - Data acquisition
  - Mapping to ontology
  - Outcome interpretation
- Background knowledge <u>unneeded</u> in analysis
  - Little sensitivity to choice of parameters / features

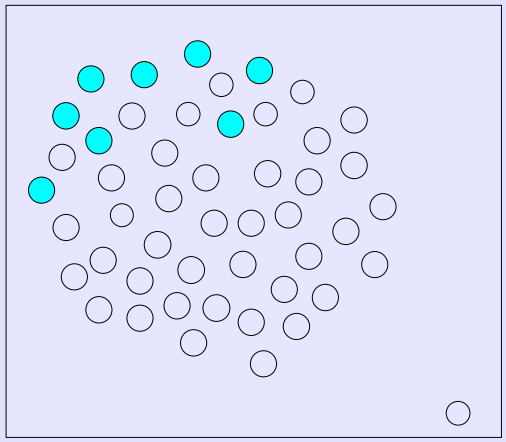


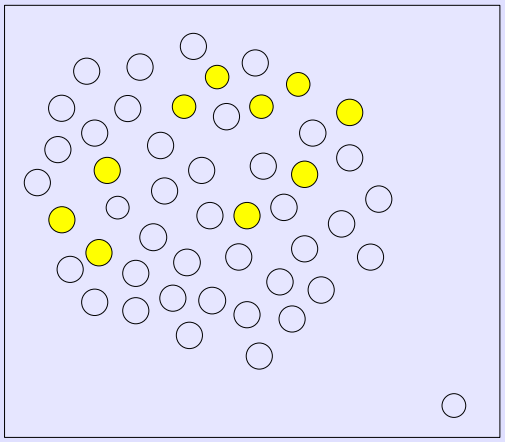
Distributed Outliers Detection

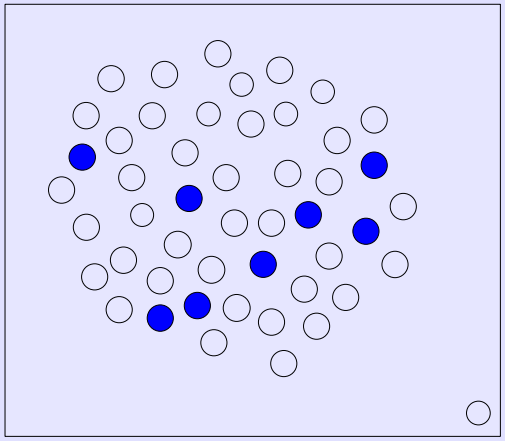
- Find the outlier

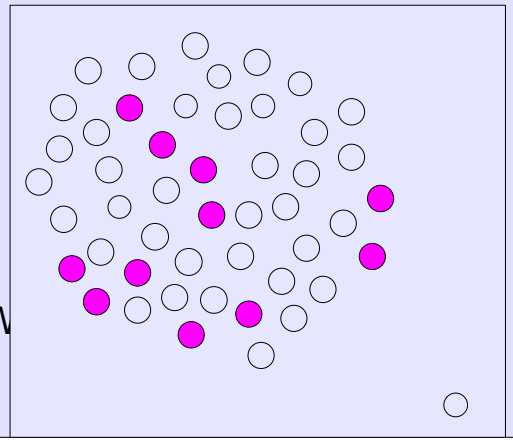


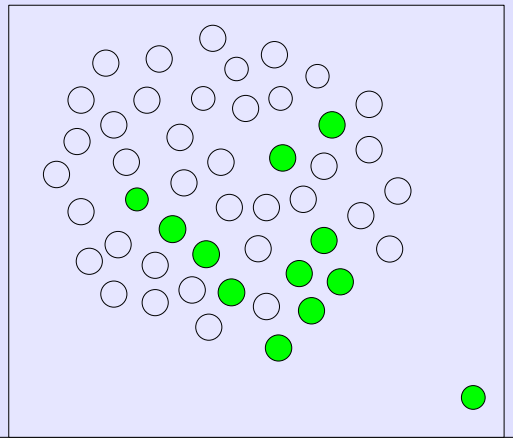
- Even though data is distributed





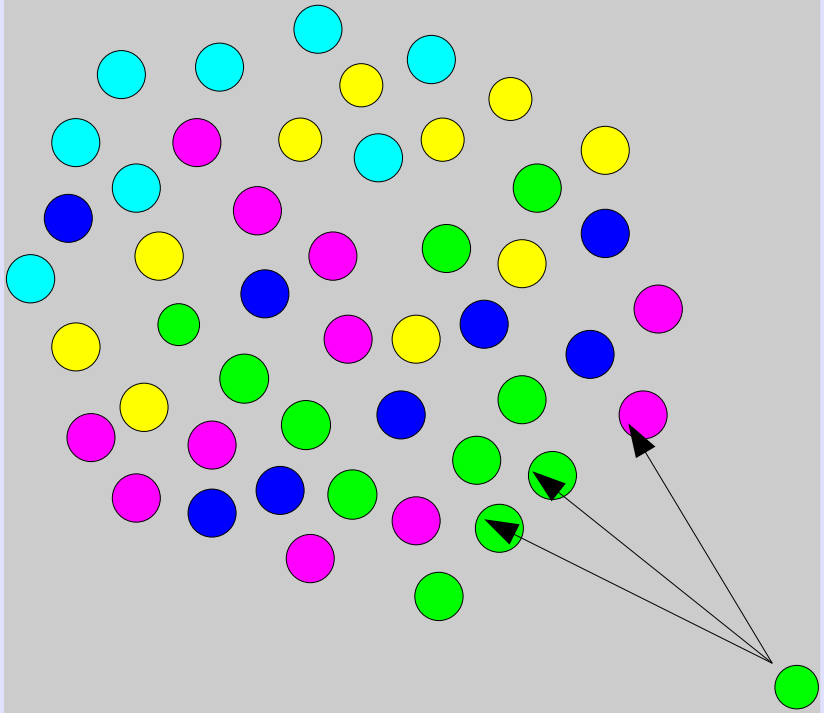


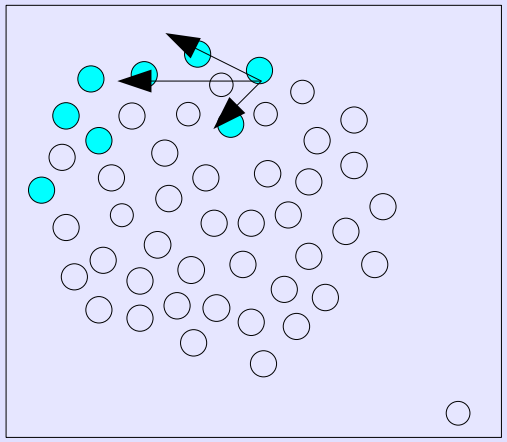


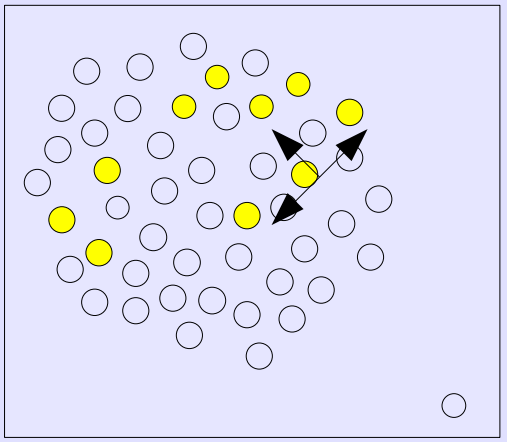


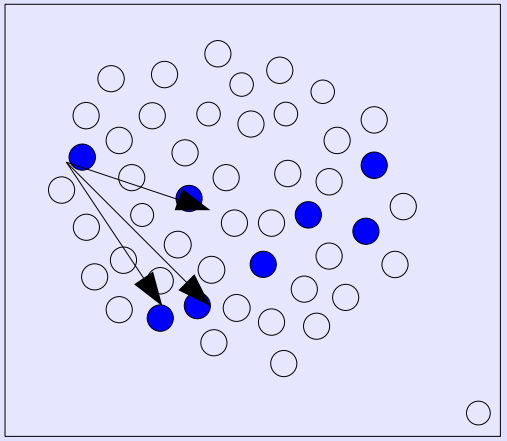
Distributed Outliers Detection

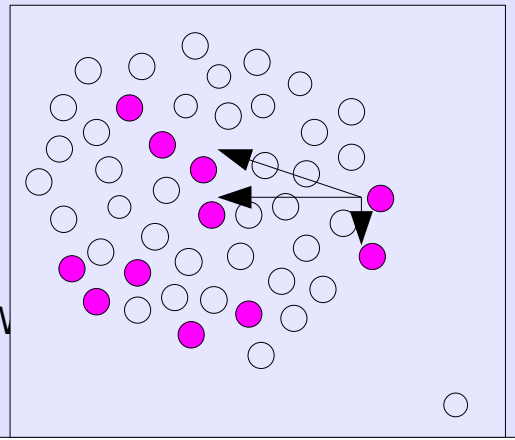
- Apart, each sees a different outlier

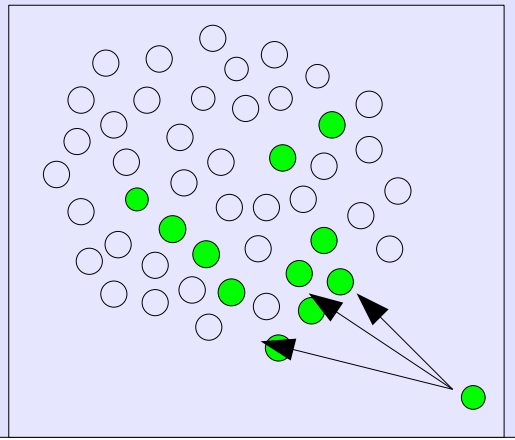






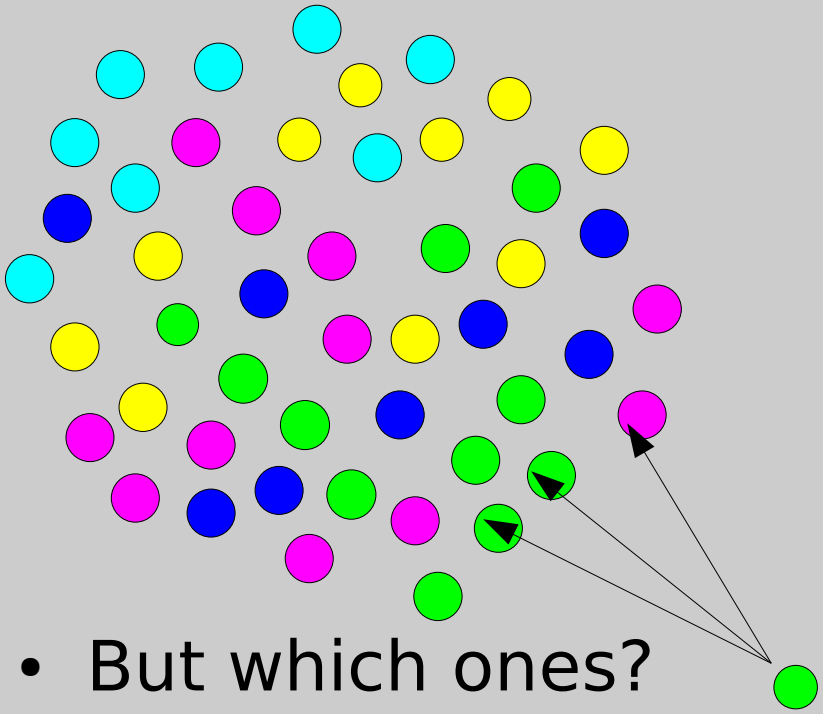




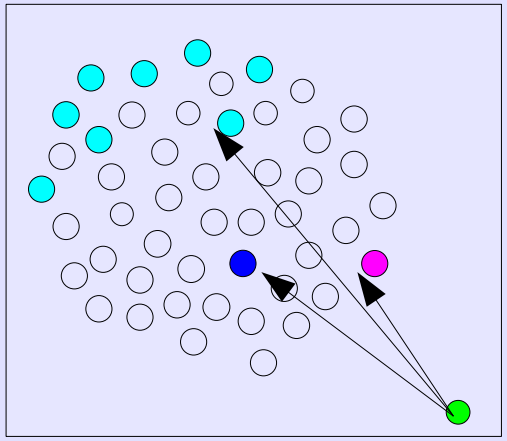


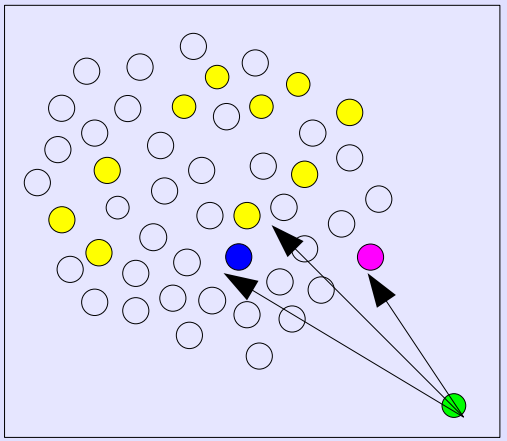
Distributed Outliers Detection

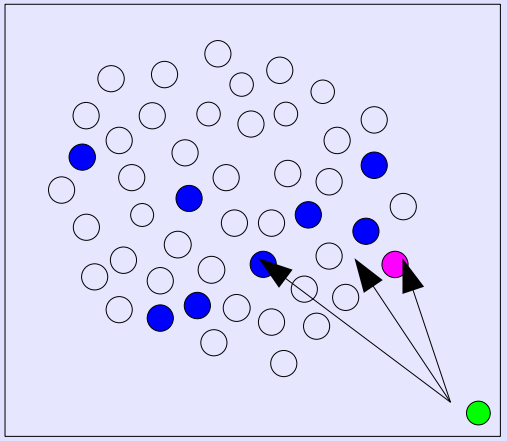
- If we just share some points

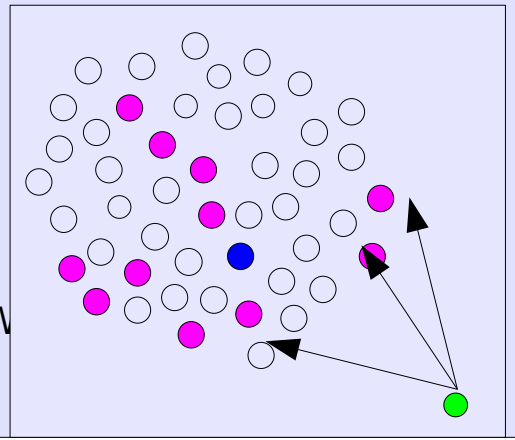


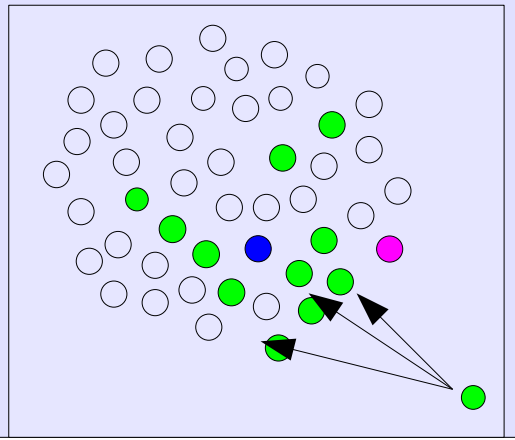
- But which ones?
  - Only interesting ones











One Theorem

- If every partition shares:
  - Would be outliers
  - Nearest neighbors of would be outliers
- Then the solution on the shared points converges to the solution on all points
- **The small print:** Correct for some, not all definitions of outliers

How the Algorithm Works

- Gradual development of a solution set
  - A set of points producing the same outcome as all points
- Take advantage of HUGE data redundancy
  - Selection of witnesses is not critical
  - For every outlier / non-outlier there are plenty of witnesses
- No sync required
- Ad hoc output
  - Dynamic data and solution

Implementation

- The procedure for choosing points to share – a job
  - Accesses a local data resource
- Shared points transferred as I/O files
- Terminating jobs spawn new jobs
  - Recursive work flow
- The whole algorithm is run in either regular or low priority

The Bottom-Line

- Executed on an operational pool
- Administrator new, a-priori, of one faulty machine
  - We found four
- Highlights:
  - The faulty machine was faulty, but for reasons other than suspected
  - A machine with hyperthreading turned on
  - A full disk
  - A run-out service

Lessons Learned

- Decentralized storage and processing is critical
  - Costs nearly nothing
  - Permits leveraging on data redundancy
- Reaffirm KD is useful for grid management
  - Shown elsewhere in large distributed systems (eBay)
  - Future research – meta-scheduling, proactive caching
- Importance of asynchronous distributed algorithms
  - Progress with whichever resource that becomes available
  - Build an ad-hoc solution which converges to the exact