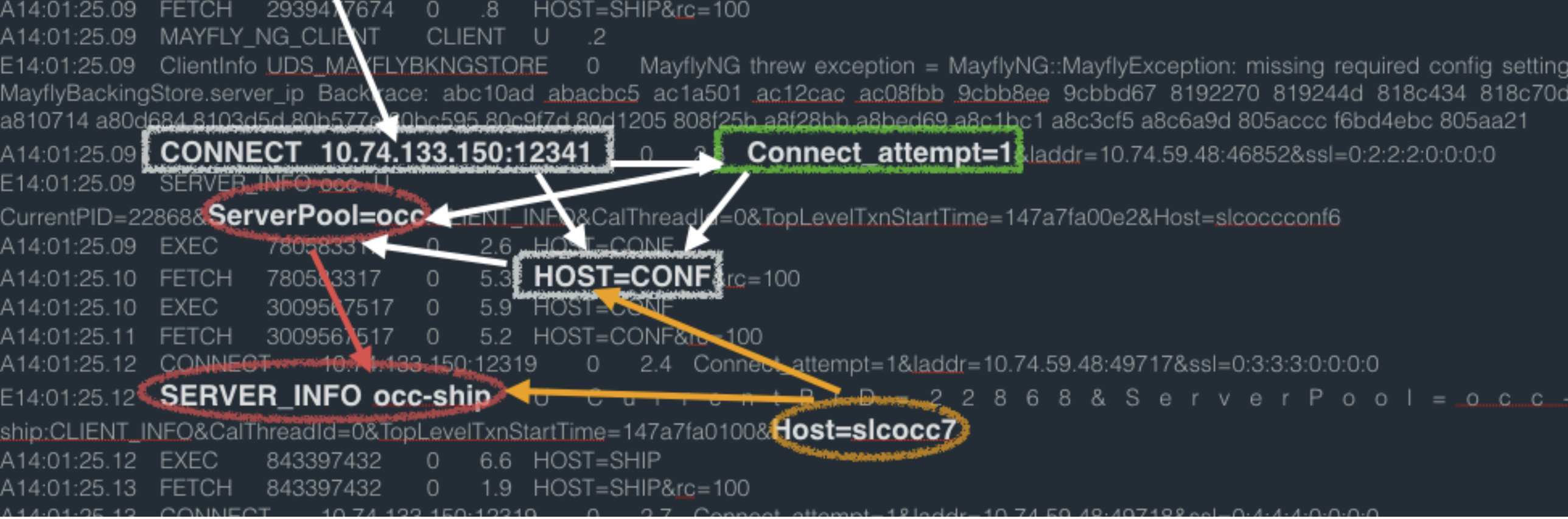


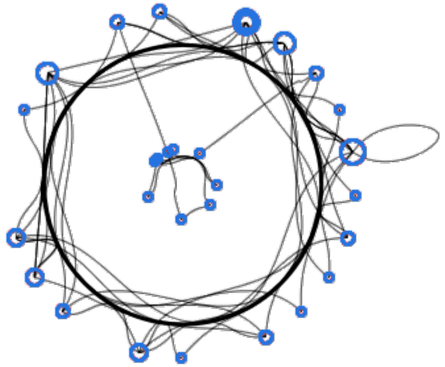
HPC IN THE HYPERSCALE ENTERPRISE:

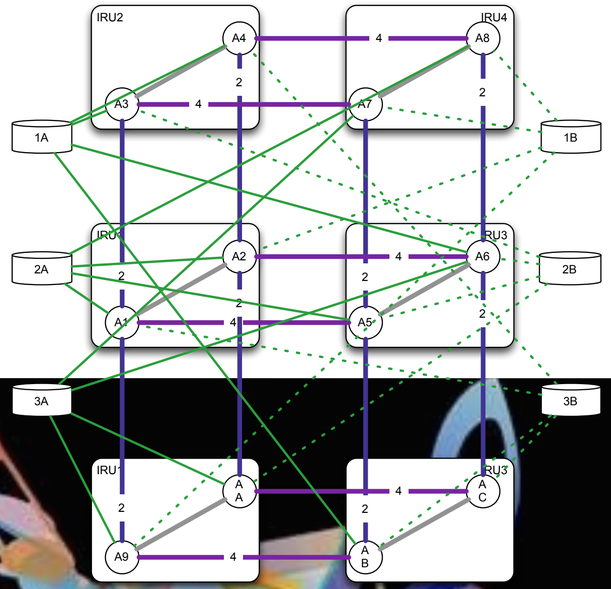
DESIGN PATTERNS AND APPLICATIONS FOR ACCELERATION

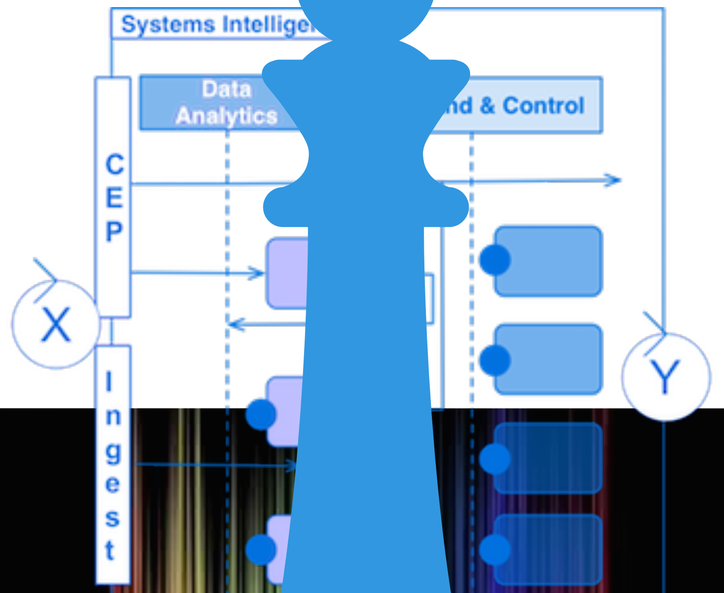
Providentia Worldwide

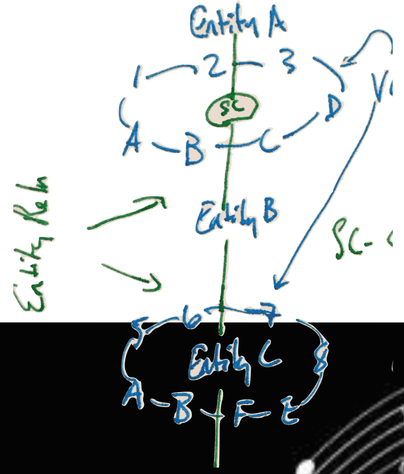
S. Ryan Quick
Principal and Co-Founder
@phaedo

Providentia Worldwide



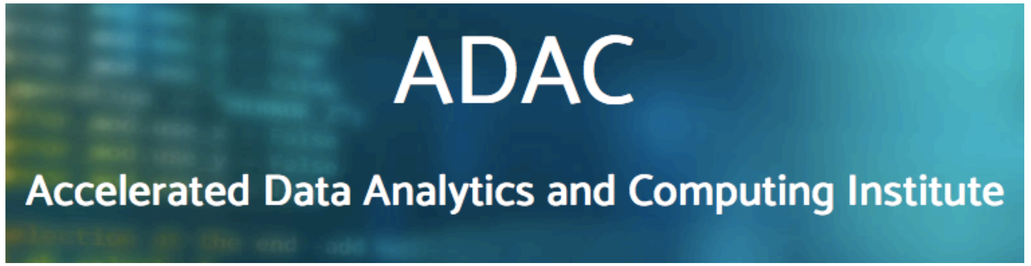






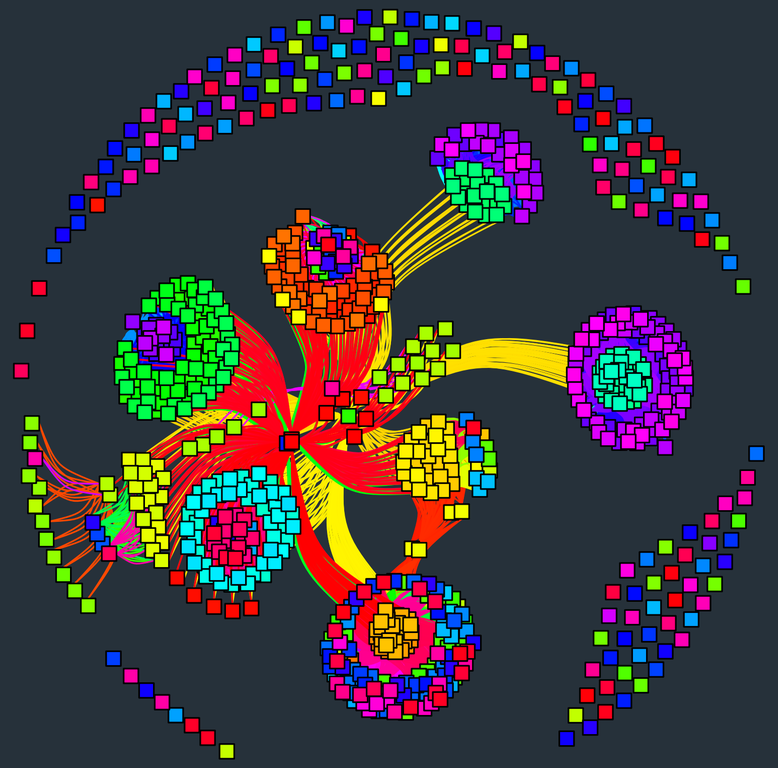
The Accelerated Data Analytics and Computing Institute has been established to explore potential future collaboration among UT-Battelle, LLC (UT-Battelle), the Swiss Federal Institute of Technology, Zurich (Eidgenössische Technische Hochschule Zürich/ ETH Zurich), and Tokyo Institute of Technology (Tokyo Tech). Consistent with their respective missions, the **Participants seek to collaborate and leverage their respective investments in application software readiness in order to expand the breadth of applications capable of running on accelerated architectures.** All three organizations manage HPC centers that run large, GPU-accelerated supercomputers and provide key HPC capabilities to academia, government, and industry to solve many of the world's most complex and pressing scientific problems.

"Purpose Of The Institute"



“We take technological artifacts to be points at which the theory-making efforts of different groups overlap, intersect, and interact.”

- Genevieve Bell, Intel @feraldata

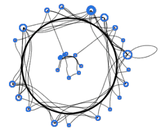


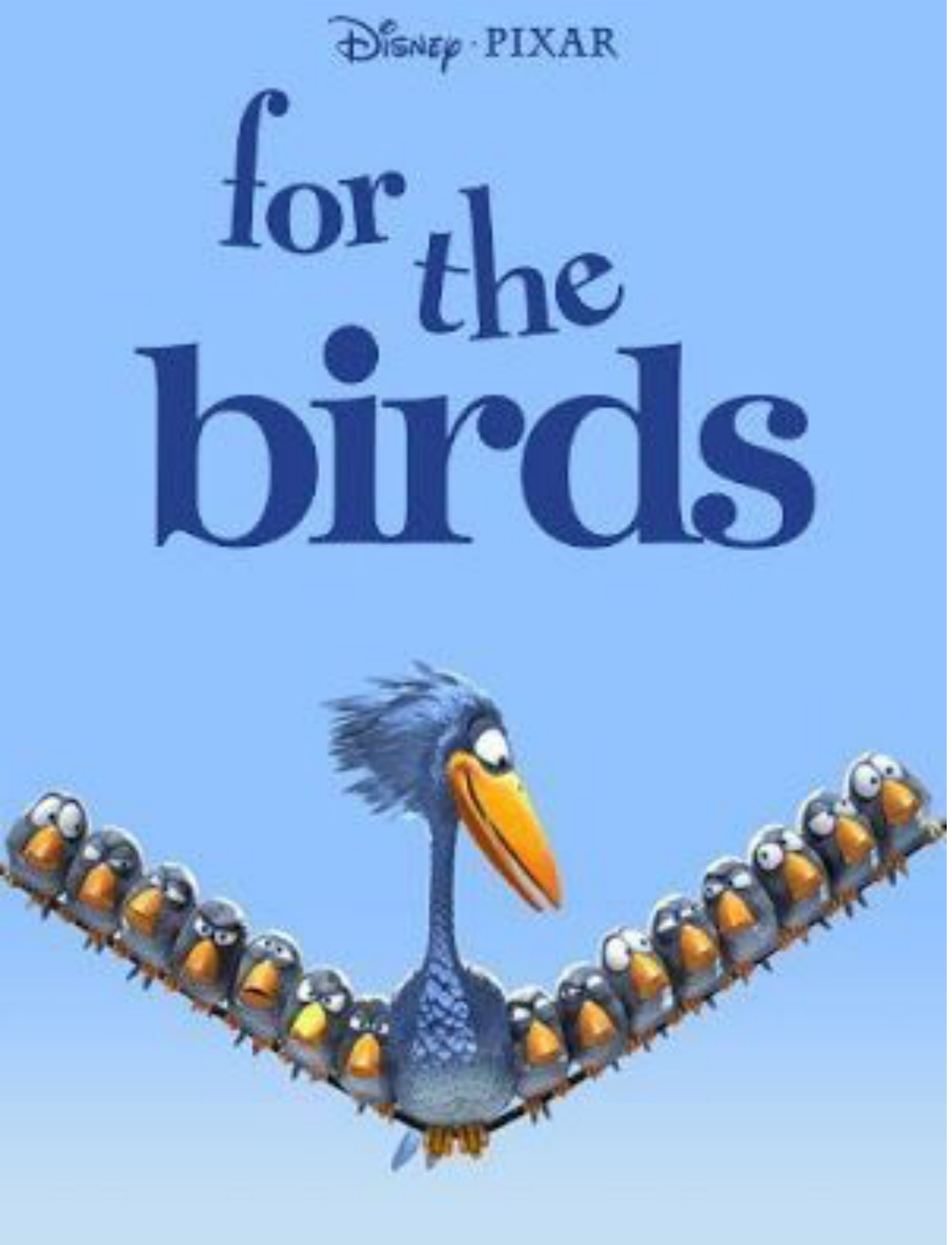
KNN, Decision Forest
56,000 dimensions
3M msgs/sec

Case 1: Map-Reduce To Shared Memory Scaleup

“Our hadoop job gets OOM-killed after day 6!’

- Map-reduce well suited for finding smaller-than-haystack needles in many haystacks (where a haystack is easily divisible amongst machines)
- Combination jobs (where the “reduce” is larger than the map) work against the core design and so must be severely segmented or risk OOM — segmentation works, but requires application-level reassembly (“Humpty Dumpty” Problem)
- Data originates elsewhere and the results are needed somewhere else (always part of larger workflow(s))

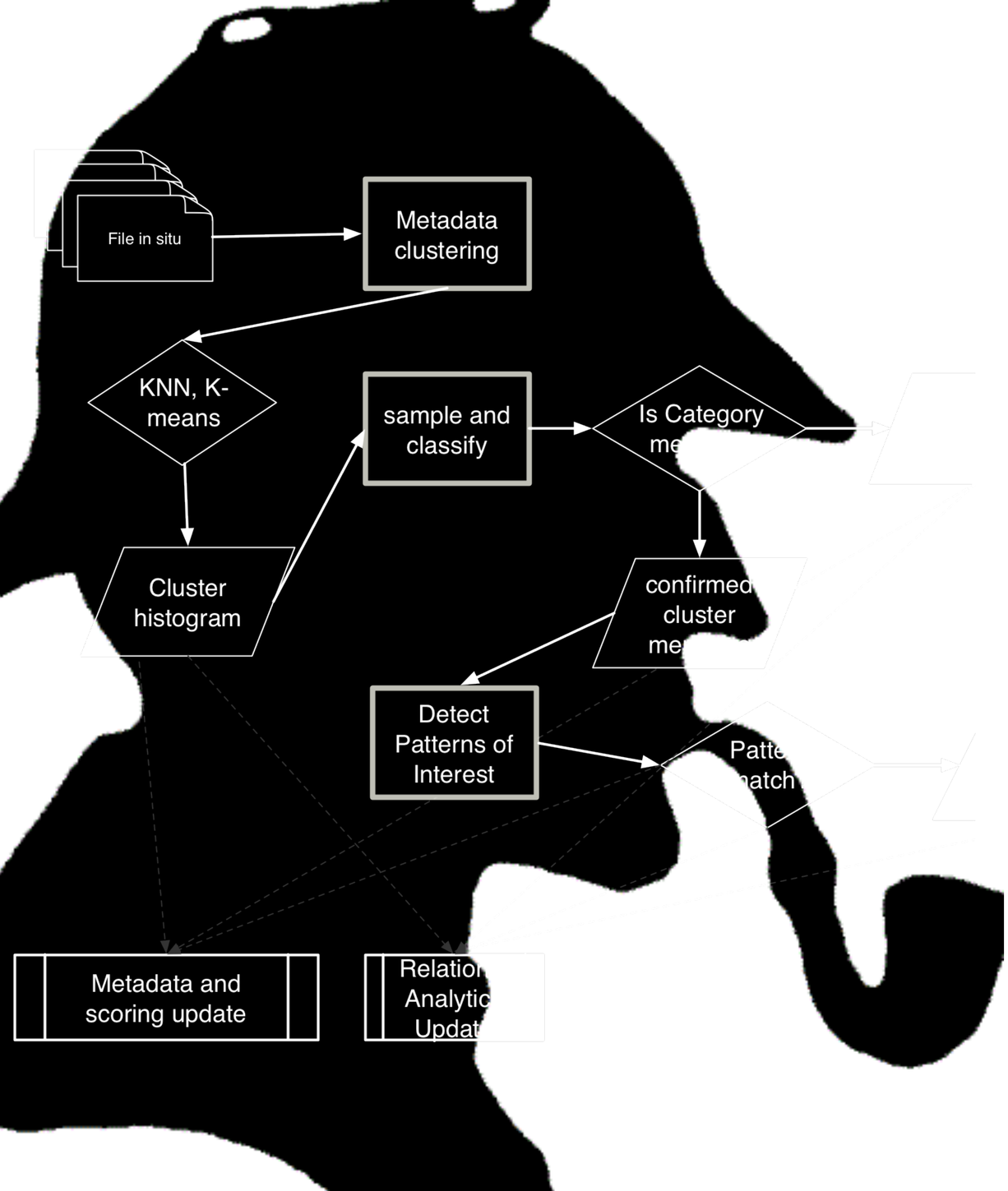




Case 1: Map-Reduce To Shared Memory Scaleup

HPC Acceleration: “Big Small Soldiers”

- Leverage large shared memory scaling for “reduce” operations
- RDMA to alleviate the implications of required data copies.
- Optimize data IO so mappers and reducers avoid internal copying
  - accelerate initial loading and “scratch IO” using memory filesystems
  - Generally: memfs > parallel fs > Object Store > HDFS/NFS/CIFS
- For ML, classification, similarity workloads, compute acceleration (GPU, DSP, TPU) are possible, but frameworks are limited
  - Need Java/Scala, Python, Go, Swift, etc as first class citizens.
  - Some progress from the DL world, but only for small set of problems.



Case 2: “Files of Interest”

Multi-component, services-based Workflow with a variety of learning methods, pattern recognition, and messaging

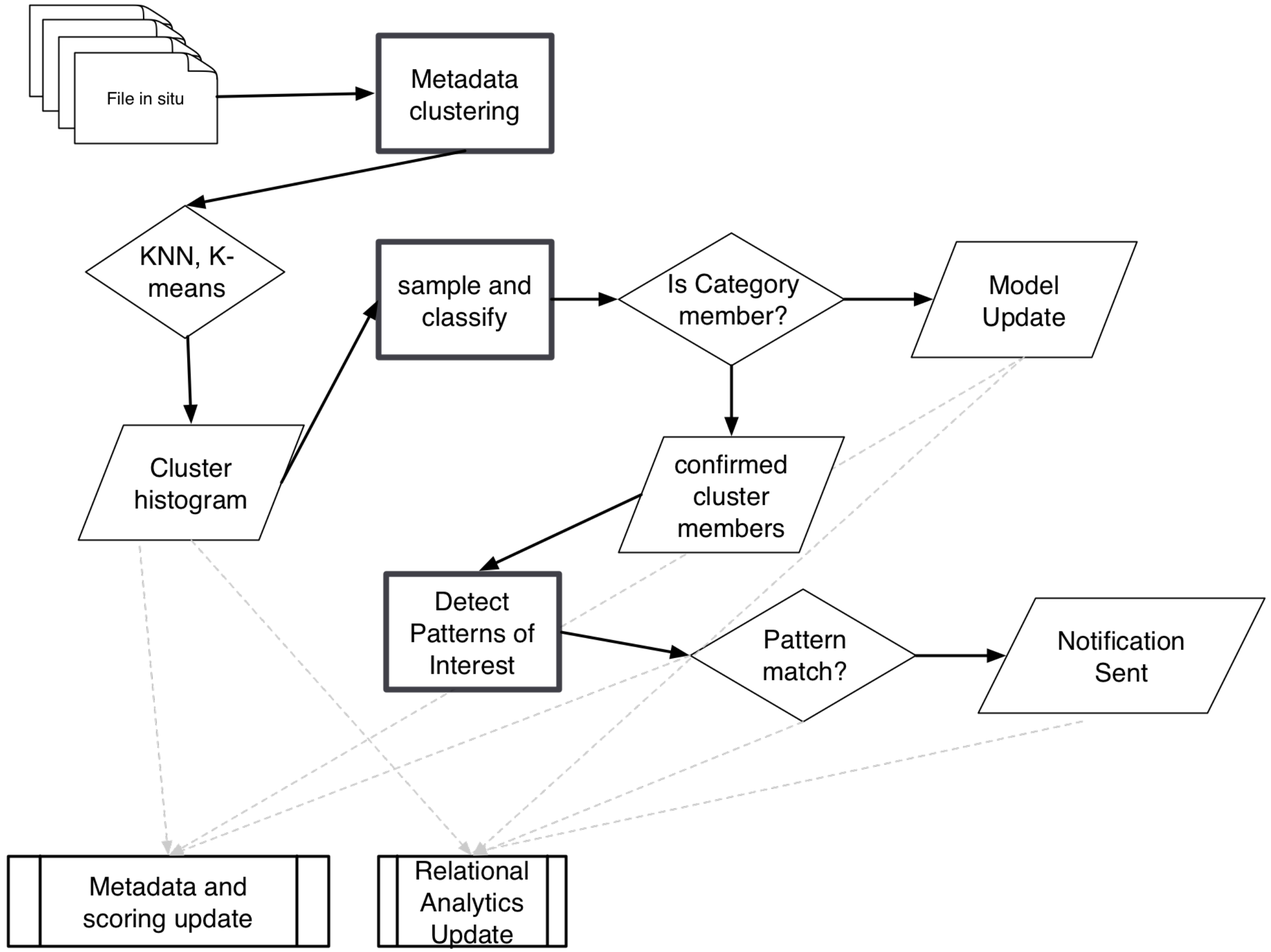
- SoA is common: components are always externalized and optimized independently. (HA + Scale)
- Messaging assumed for coordination, data movement and routing, schema independence, pub/sub scaling, etc.
- Acceleration opportunities abound, but usually the TTM for a feature will win over performance, so we need a performant functional programming system. (Deep Learning is ahead here, but doesn’t work for all problems)
- Event ordering is not important here. General workflow sequence is good enough...

“Files of Interest”

HPC Acceleration: ML, DL, Pattern Recognition, Event Streaming

Given an organized structure of files

- determine groups of similar files, without opening the file itself
- classify a statistically relevant sample of each group to confirm validity of grouping
- at regular intervals, scan for patterns of interest against all files, using class-specific patterns
- send event notifications to command and control systems when patterns of interest are detected, including enough insight for those systems to handle the event

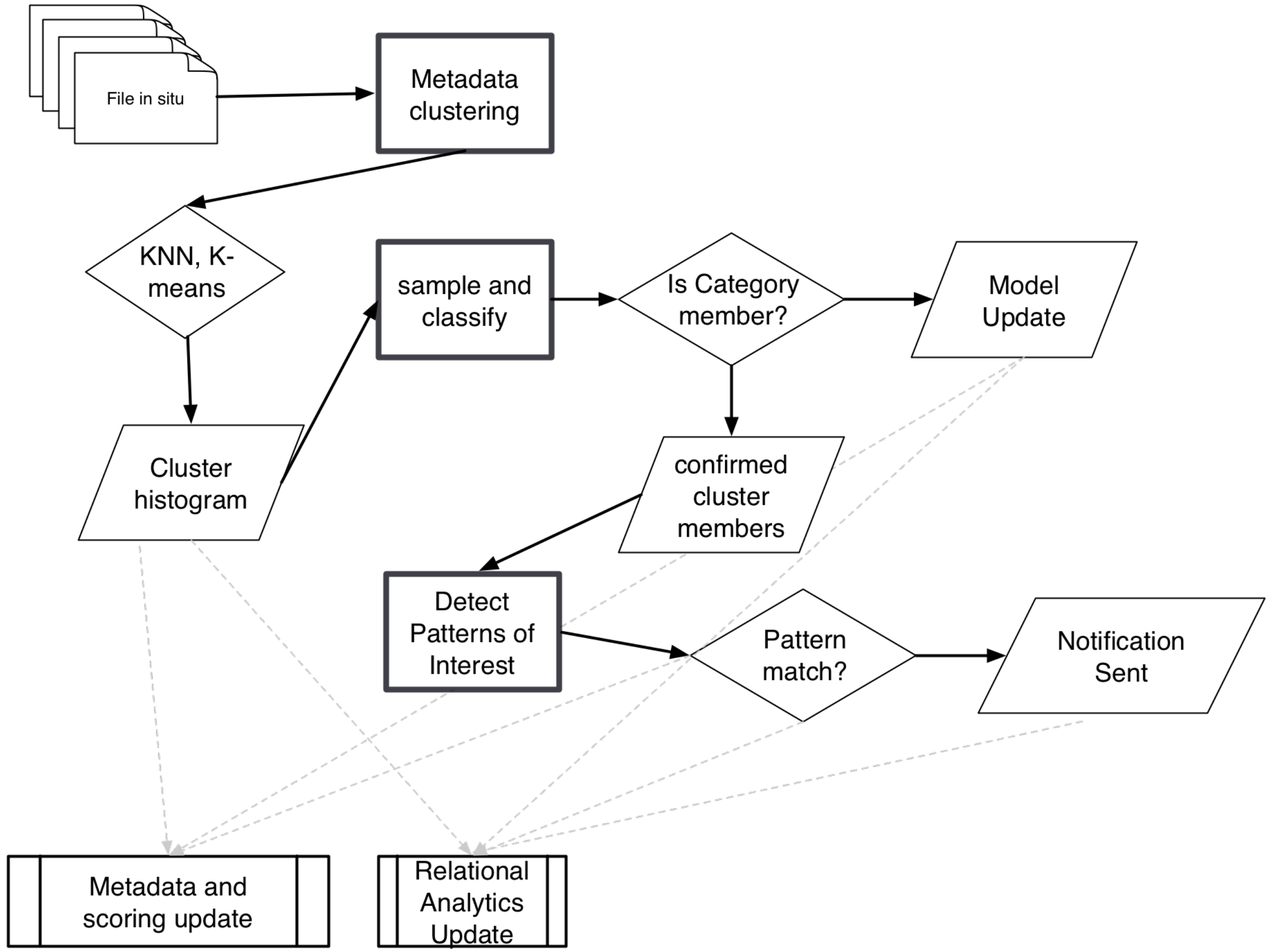


“Files of Interest”

HPC Acceleration: ML, DL, Pattern Recognition, Event Streaming

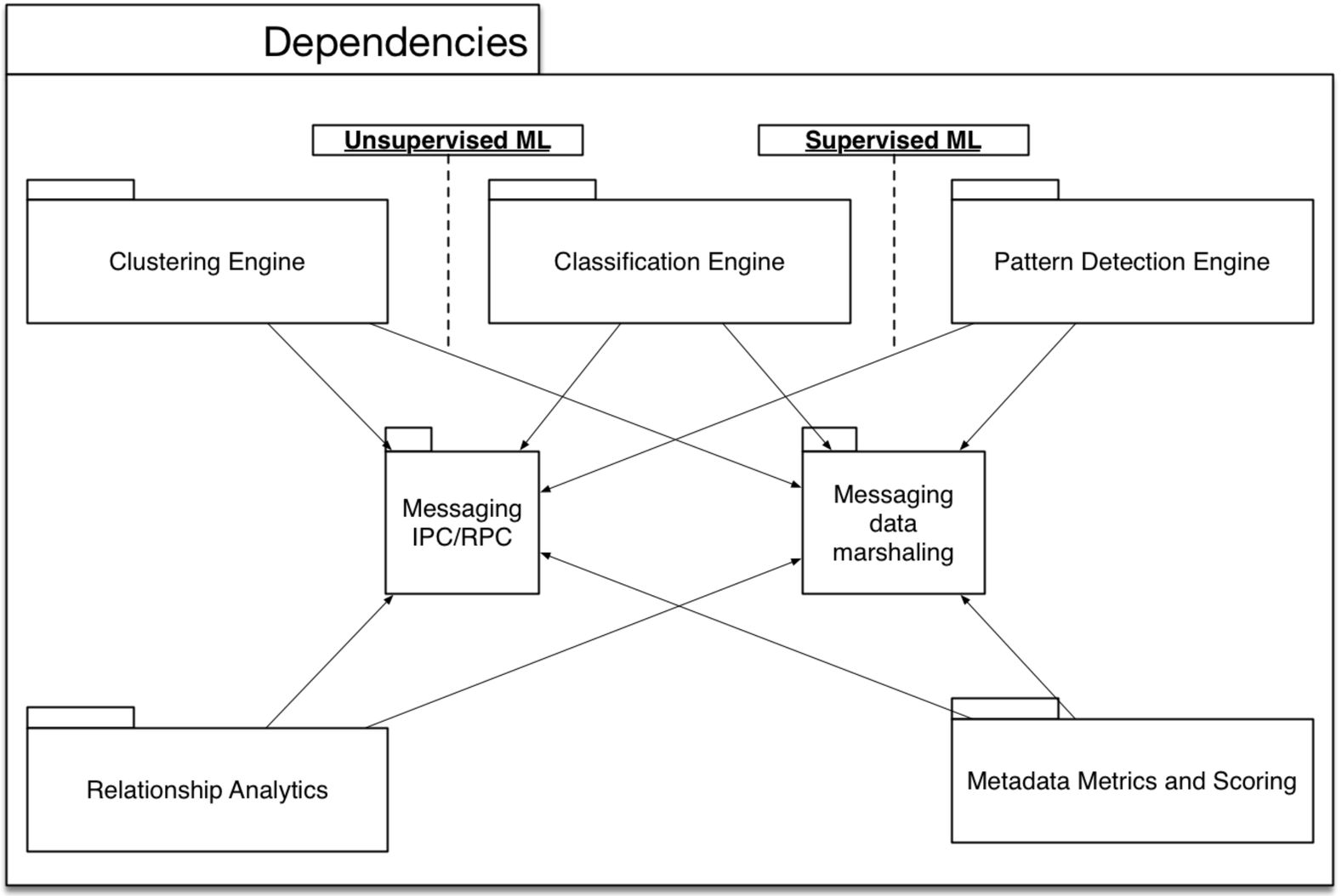
Technology Components:

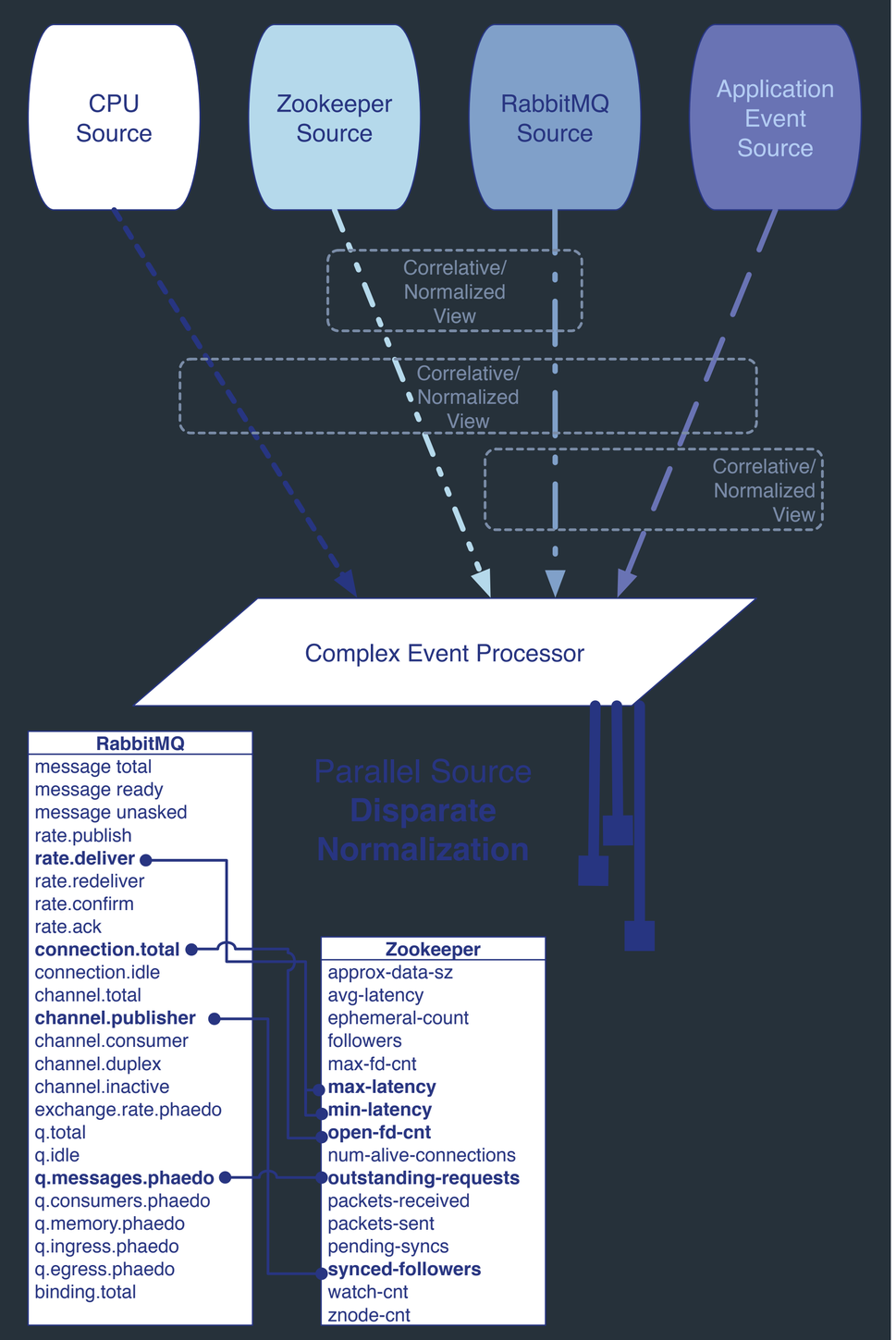
- Messaging Middleware
- Metadata Schema (relational)
- Metadata Relationships (graph)
- Unsupervised Learning (Clustering)
- Supervised Learning (Classification)
- Deep Learning (Pattern Determination)
- Pattern/Anomaly Recognition
- Event Generation



“Files of Interest”

HPC Acceleration: ML, DL, Pattern Recognition, Event Streaming

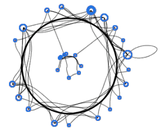


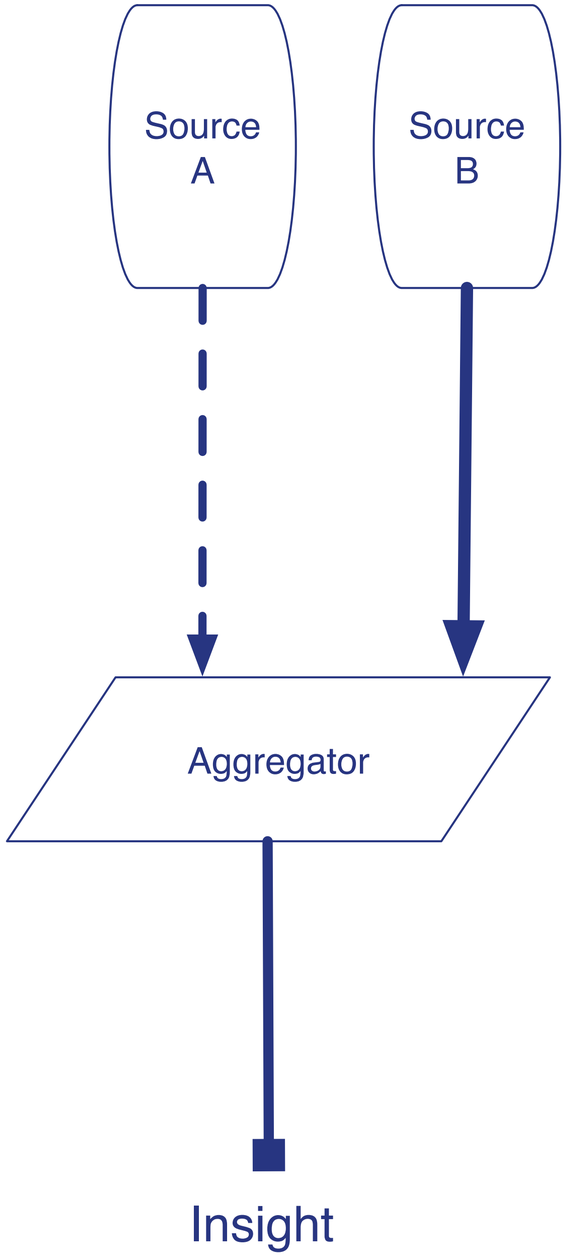


Case 3: Real-Time Stream Analytics

Compound Messaging Paradigms For Parallel Source, Parallel Stream, Affinity Analytics

- One of the best opportunities for acceleration on a variety of resource vectors
  - in-network
  - in-memory
  - compute
  - semantics and ontology
  - ordering, prioritization, delivery optimization





Simple Approaches

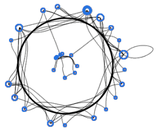
Messaging: Source Aggregation

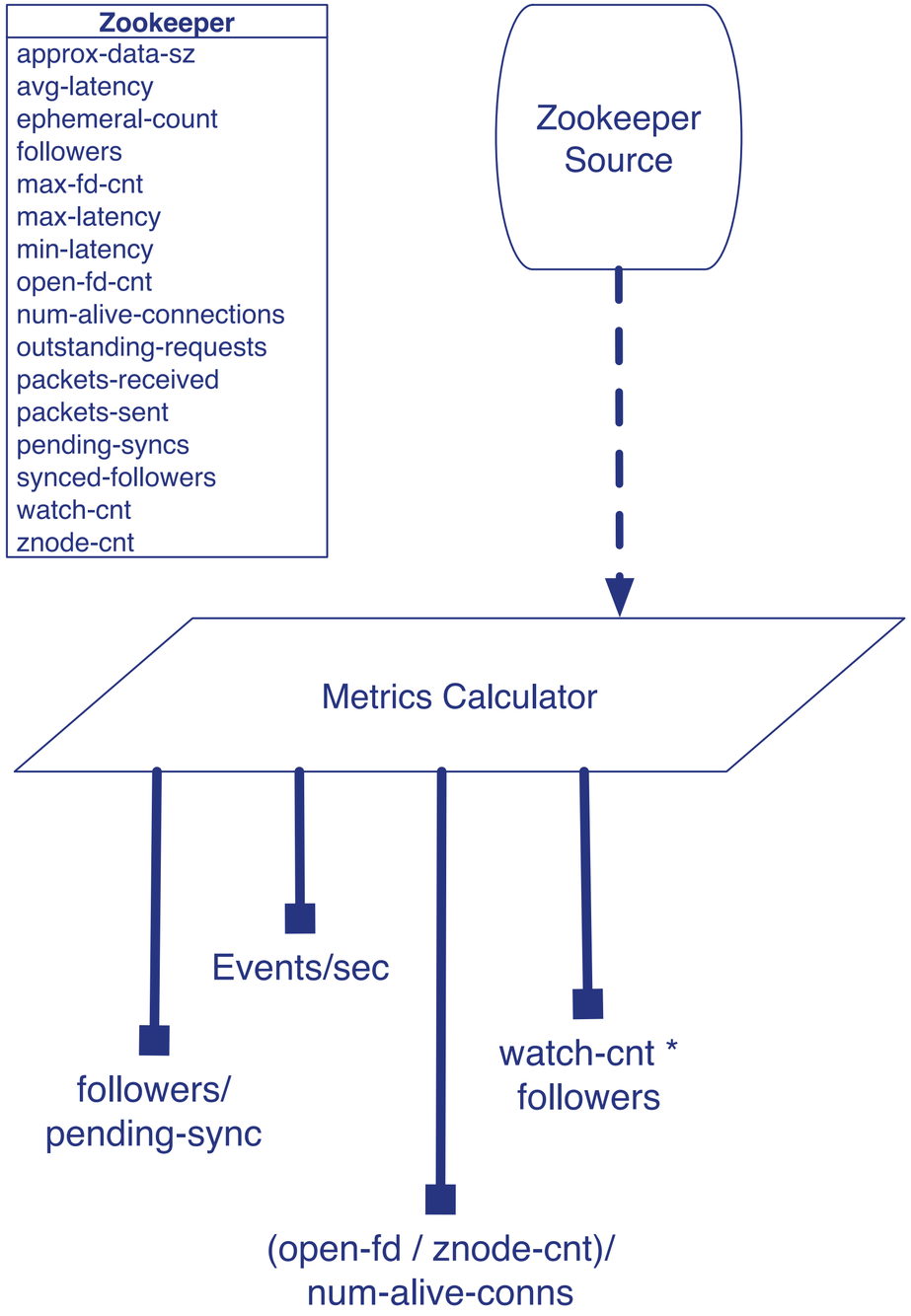
Aggregation

Event Statistics

Atomic Pattern Recognition

- Stream sources are combined in an aggregation application.
- Output is derived insight based on both sources
- Use Case Example: CPU performance related to TCP Connections
  - A: CPU idle % every 30s
  - B: TCP connections (incoming) [Event Driven]
  - INSIGHT: TCPCONNS/IDLE %





Simple Approaches

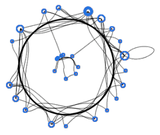
Messaging: Event Statistics Calculations

Aggregation

Event Statistics

Atomic Pattern Recognition

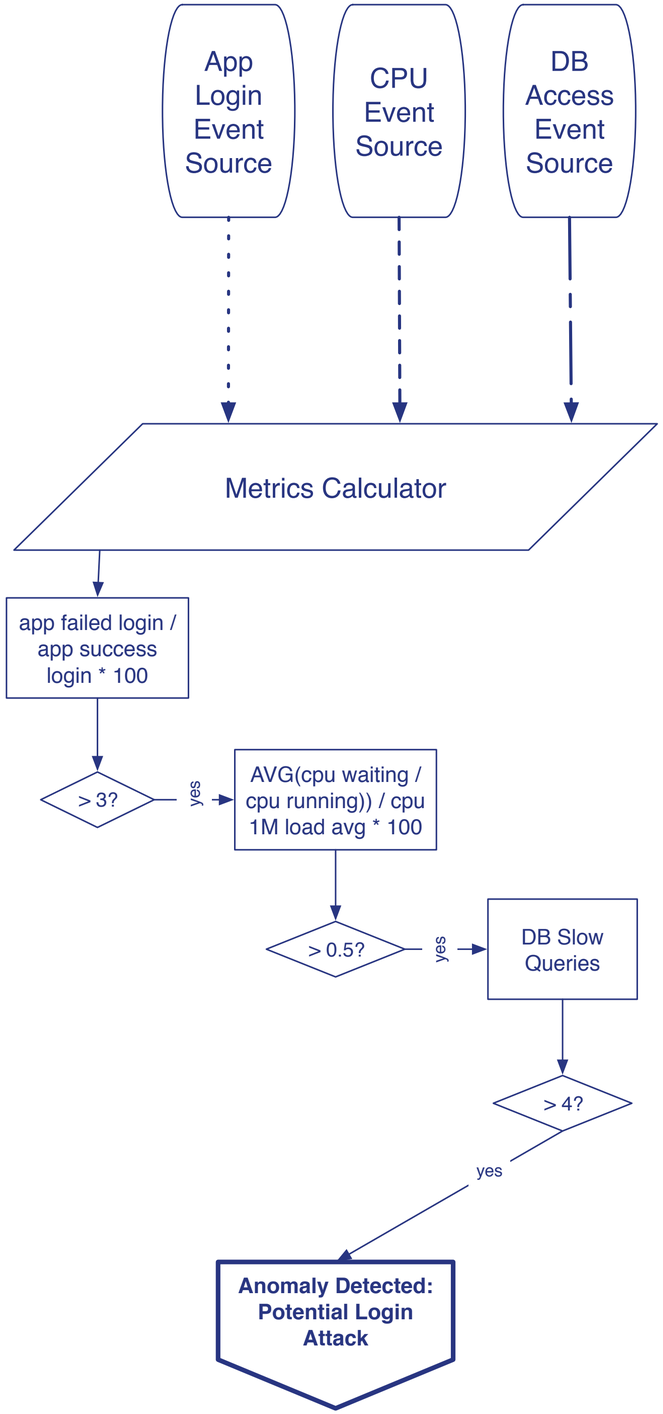
- Numerical/Categorial calculations based on data contained within the source datum/event
- Output insight effectively introduces new sources, generally numerical/gauged.
- Use Case Example: Watched-Files-Per-Active-Consumer output as new stream source
  - INSIGHT: watch-cnt (value per event) * synced-followers (value per event)



Simple Approaches

Messaging: Multi-Source Atomic Pattern Recognition

Available Source Fields
app login r/sec
app successful login r/sec
app failed login r/sec
cpu 1m load avg
cpu 5m load avg
cpu 15m load avg
cpu blocked proc cnt
cpu running proc cnt
cpu waiting proc cnt
cpu user %
cpu idle %
cpu system %
cpu io wait %
db active queries
db slow queries
db selects
db updates
db deletes
db rows fetched
db table locks held
db row locks held

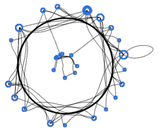


Aggregation

Event Statistics

Atomic Pattern Recognition

- Simple thresholds within the event itself
- Correlation can be within a single source, or across disparate sources
- Represented as “waterfalling” but this depends on frequency and is really just easier for us to read, the operations are parallel and stateless (in this approach)
- Use Case Example: Output Potential-Login-Attack events



Compound Approaches

Messaging: Affinity Source Collection

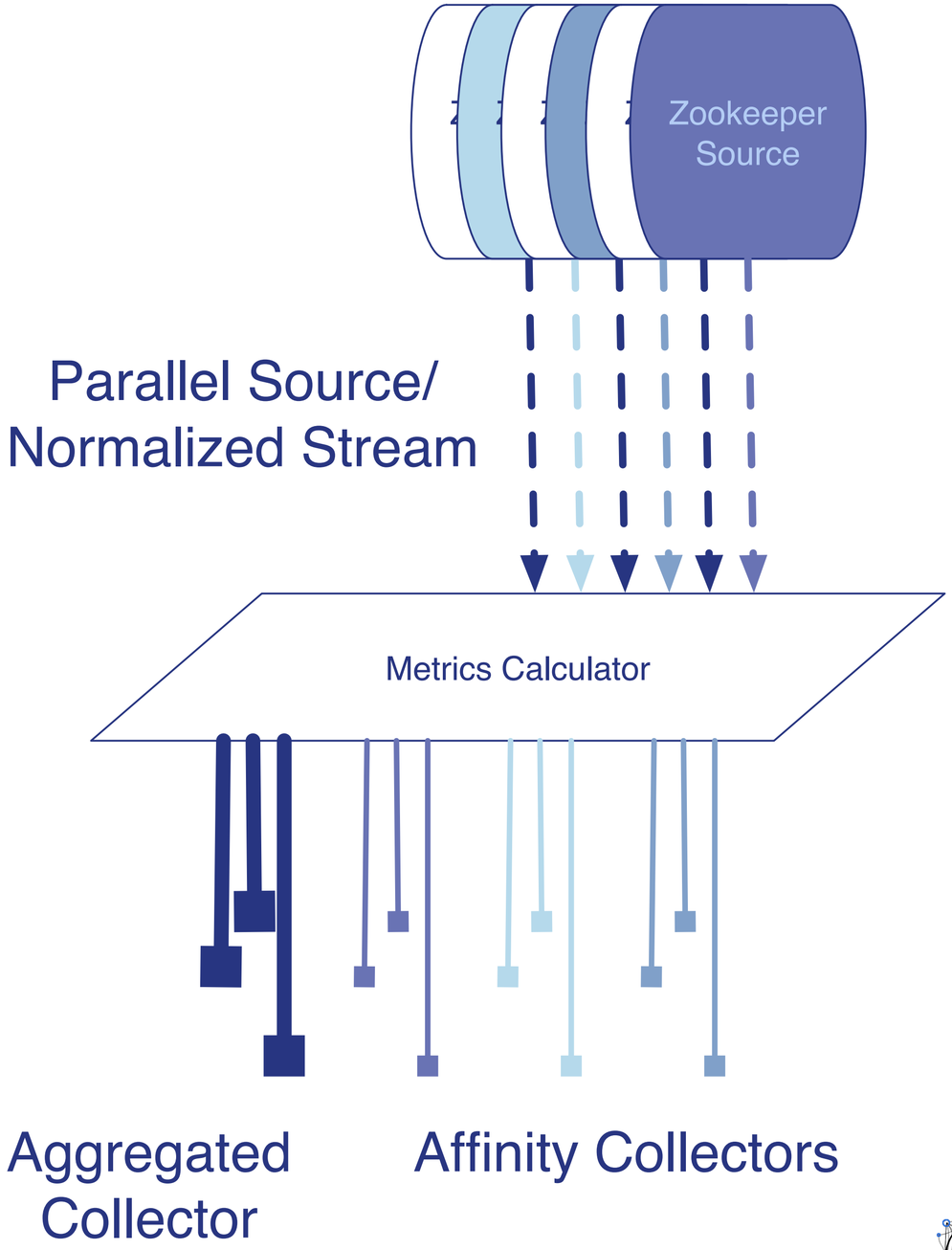
Affinity + Simple Case

Stream + Augmented Datasource

Parallel Stream

Frequency-Shifted Stream

- Given parallel publishers for single source schemas, affinity refers to collating events by
  - publisher
  - schema
  - both
- Can be implemented automatically based on other simple cases
- Use Case Example: “Person of Interest”, “Behavior of Interest”
  - Collate data by publisher once an anomalous event is triggered by a simple approach
  - Collate all like-schema sources to watch “pool behavior”



Compound Approaches

Messaging: Parallel Source And Datasource Augmentation

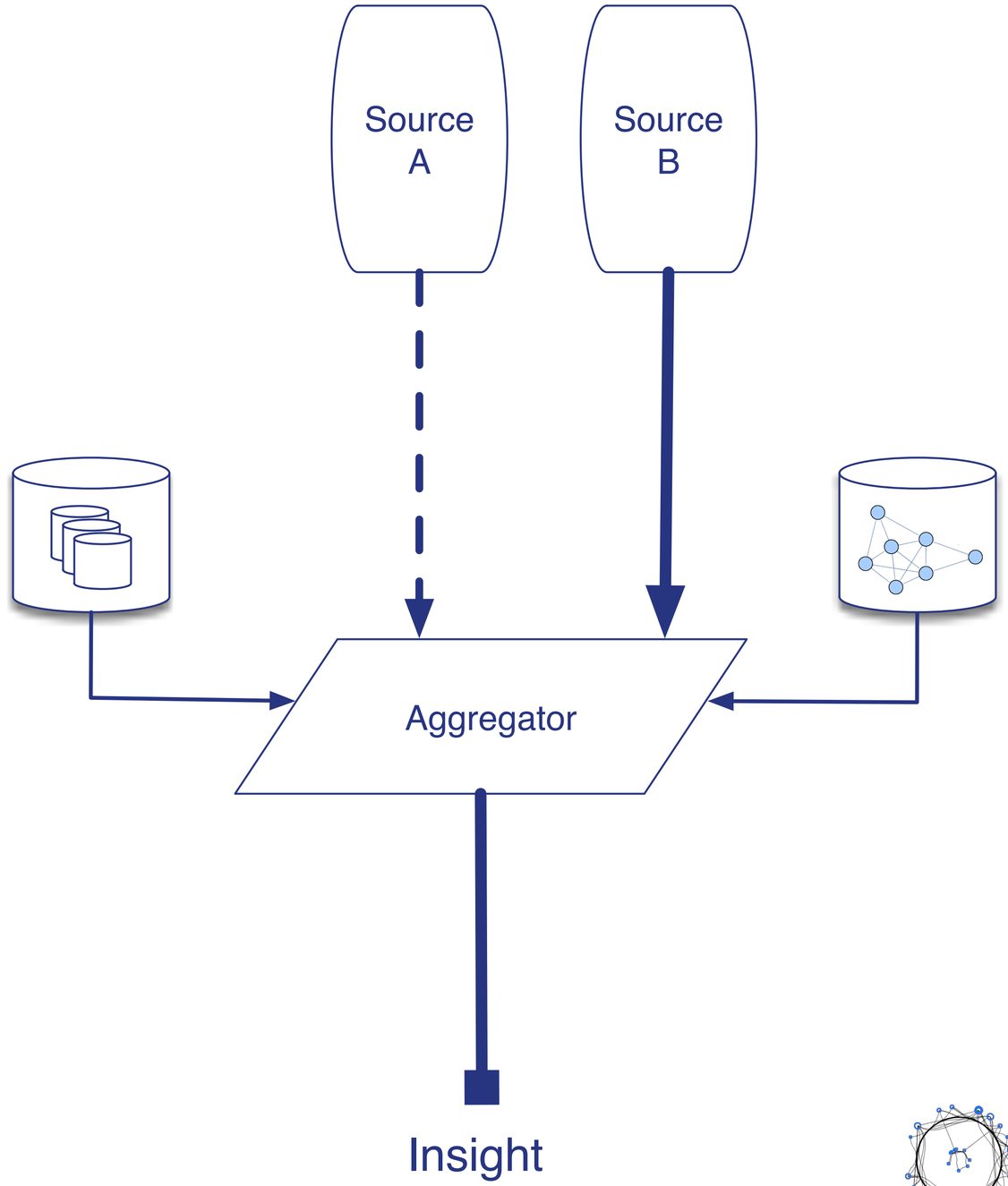
Affinity + Simple Case

Stream + Augmented Datasource

Parallel Stream

Frequency-Shifted Stream

- Source data is augmented by
  - additional sources (alternate schema)
  - additional data sources (RDBMS, GraphDB, KV, Cache, etc)
- Used in cases where information on the wire requires additional context, culling, augmentation to provide insight
- Use Case Example: Network Detection
  - Event Source provides transaction details, network actors
  - RDBMS provides known-network attributes
  - Graph DB provides existing actor-network
  - Aggregator determines similarity score that the current event is a particular network type



Compound Approaches

Messaging: Multi-Source, Multi-Stream (Stream Tensor Normalization)

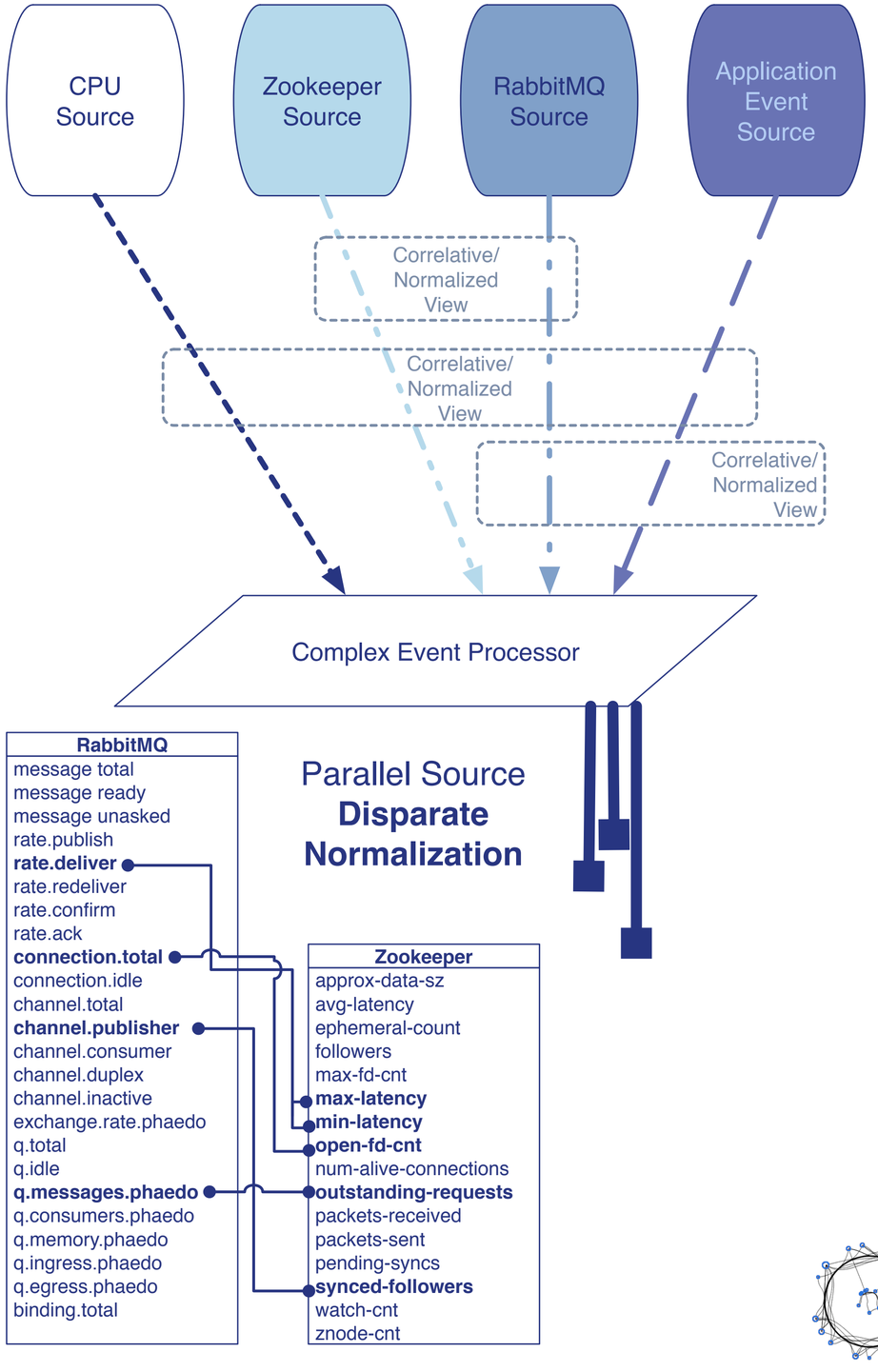
Affinity + Simple Case

Stream + Augmented Datasource

Parallel Stream

Frequency-Shifted Stream

- "Correlative/Normalized View": Similar to a SQL "join" concept, we relate data fields in disparate stream sources
- Requires frequency mapping (sliding windows, state management, etc.)
- Use Case Example: Messaging System and Zookeeper filesystem relationships
  - vector time (event/observation based)
  - incoming/outgoing pipeline relationships
  - actor mapping
  - filesystem/messaging performance



Compound Approaches

Messaging: Frequency-Shifting And Ordering

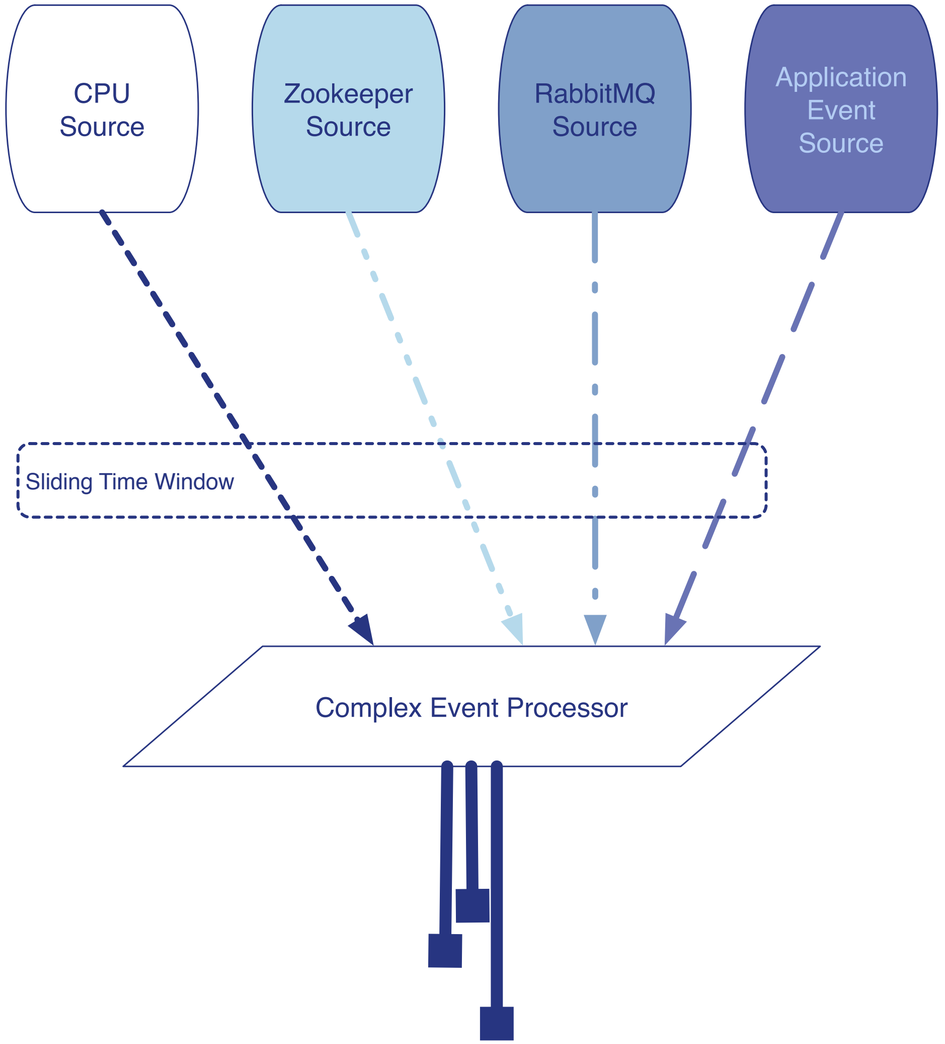
Affinity + Simple Case

Stream + Augmented Datasource

Parallel Stream

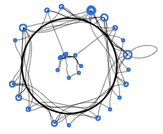
Frequency-Shifted Stream

- Not a simple problem, and is usually where the "it's easier to just do this in situ" argument comes up.
- Most sources do not publish at the same interval. To handle this we need a variety of techniques (some examples):
  - sliding time windows
  - state management (value looping)
  - relevancy-offset clocks (determined by "master events")
  - store and forward
- Use Case Example: Application Environmental CPU Impact
  - CPU published on time interval, leverage value looping
  - Application is event-driven, it's the master.



Parallel Source
Disparate Frequency

CPU	Zookeeper	RabbitMQ	Application
event_duration_ms	event_duration_ms	event_duration_ms	event_duration_ms
event_timestamp_orig	event_timestamp_orig	event_timestamp_orig	event_timestamp_orig
observed_timestamp	observed_timestamp	observed_timestamp	observed_timestamp
observation_latency	observation_latency	observation_latency	observation_latency



“Centralized Messaging”

200+ Gbs
Ethernet for datacenter interchange connectivity

800+Gbs
Infiniband connectivity
for internal communications
+ RDMA
+ Shared Memory
+ 3 Tiers of Persistence (see inset)

Datacenter

Private Cloud

Public Cloud

Third Party

TIBCO

Qpid

ActiveMQ

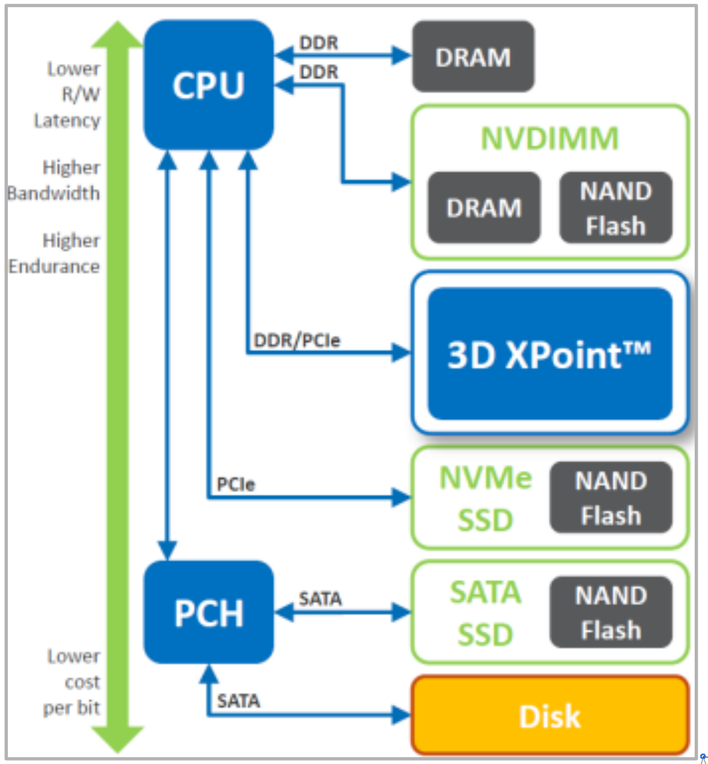
Messaging Node A

Messaging Node B

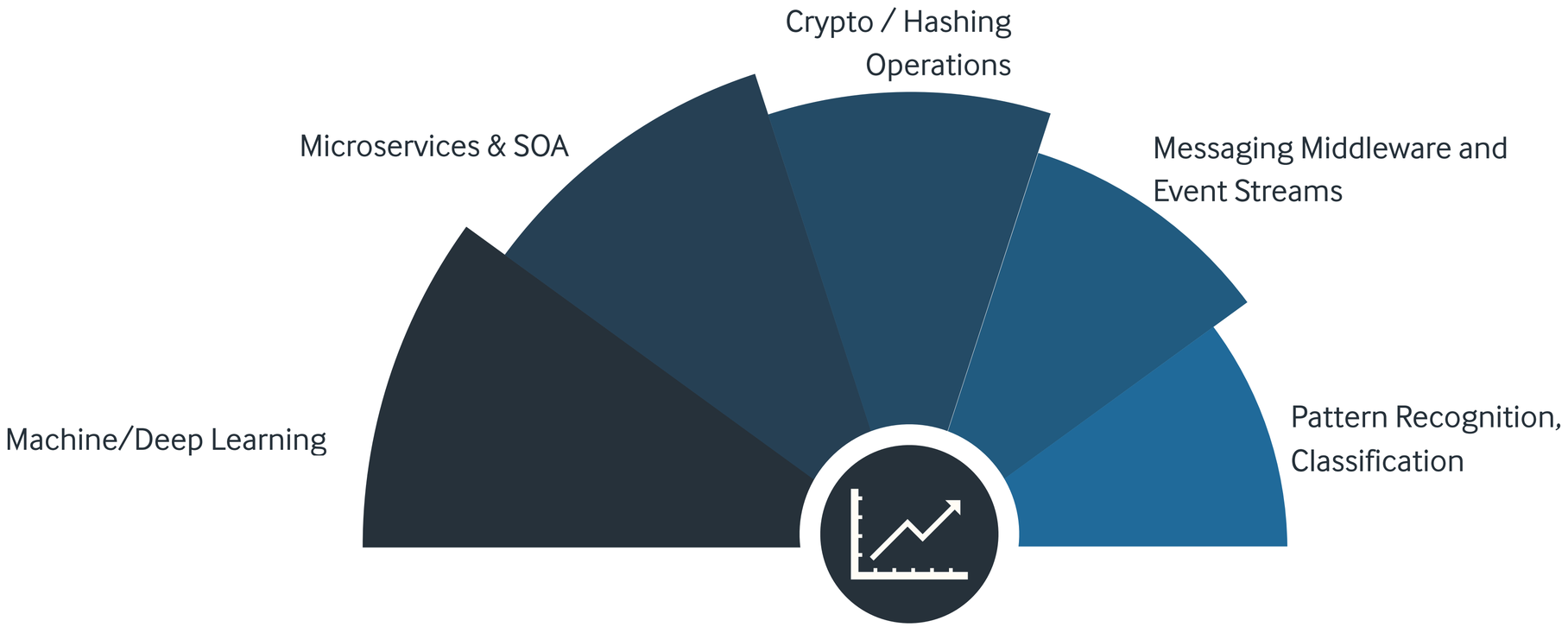
Messaging Node C

Messaging Node D

Apache Camel

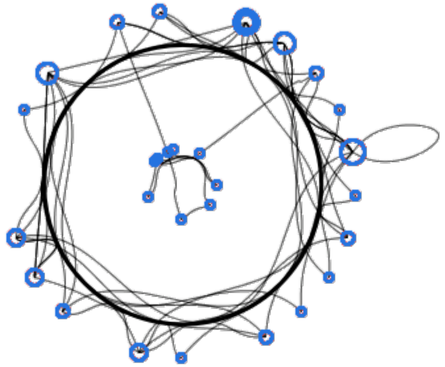


Hyperscale Acceleration Opportunities



HPC and Hyperscale will merge, now is the time to optimize and accelerate. But we accelerate the best of what they already do, not force paradigm change

Providentia Worldwide



Questions?

@providentia_ww

solutions@providentiaworldwide.com