Graph Analytics: Complexity, Scalability, and Architectures

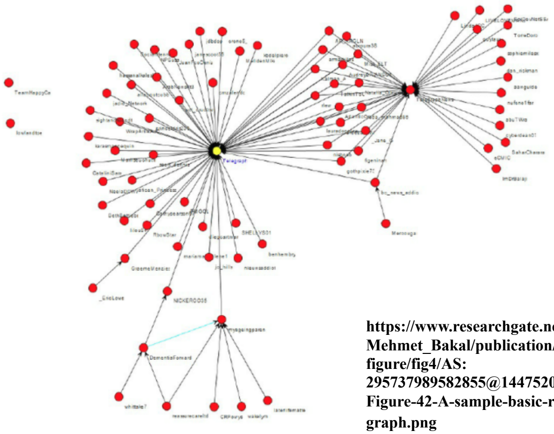
Peter M. Kogge
McCourtney Prof. of CSE
Univ. of Notre Dame
IBM Fellow (retired)

Thesis

- Graph computation is increasing
- To date: most benchmarks are batch
- Streaming becoming more important
- This talk: Combine batch and streaming
- Emerging architectures have real promise

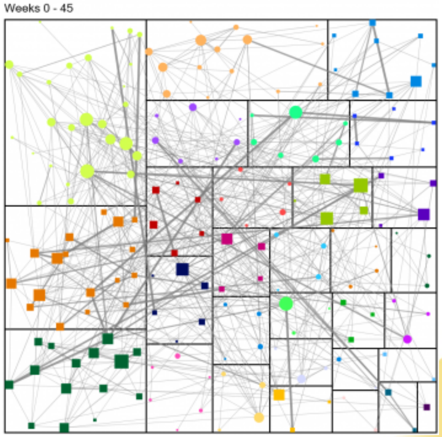
Graph Kernels and Benchmarks

Graphs



https://www.researchgate.net/profile/Mehmet_Bakal/publication/266968024/figure/fig4/AS:295737989582855@1447520839376/Figure-42-A-sample-basic-retweet-graph.png

- Graph:
  - Set of objects called **vertices**
  - Set of links called **edges** between vertices
  - May have "properties"
- Graph computing of increasing importance
  - Social Networks
  - Communication & power networks
  - Recommendation systems
  - Genomics
  - Cyber-security



http://icensa.com/sites/default/files/styles/research_image/public/Unknown.png?itok=HfBBjbJK

Classes of Graph Computation

- Characteristics of individual vertices
  - E.g. “properties” such as degree
- Characteristics of graph as a whole
  - E.g. diameter, max distance, covering
- Characteristics of pairs of vertices
  - E.g. Shortest paths
- Characteristics of subgraphs
  - E.g. Connected components, spanning tree
  - Similarities of subgraphs, …

Classes of Application Computations

- **Batch**: function applied to entire graph of major subgraph as it exists at some time
- **Streaming**:
  - Incoming sequence of small-scale updates
    - New vertices or edges
    - Modification of a property of specific vertex or edge
    - Deletions
  - Sequence of localized queries

Current Benchmark Suites

Kernel	Kernel Class						Benchmarking Efforts										Outputs					
	Connectedness	Path Analysis	Centrality	Clustering	Subgraph Isomorphism	Other	Standalone	Firehose	Graph500	GraphBLAS	Graph Challenge	Graph Algorithm Platform	HPC Graph Analysis	Kepner & Gilbert	Stinger	VAST	Graph Modification	Compute Vertex Property	Output Global Value	Output O(1) Events	Output O(|V|) List	Output O(|V|^k) List (k>1)
Anomaly - Fixed Key						X		S												X		
Anomaly - Unbounded Key						X		S												X		
Anomaly - Two-level Key						X		S												X		
BC: Betweeness Centrality			X							B		B		B	S			X				
BFS: Breadth First Search	X								B	B		B	B	B	B			X			X	
Search for "Largest"						X						B									X	
CCW: Weakly Connected Components	X												B	B	S			X			X	
CCS: Strongly Connected Components	X												B	B							X	
CCO: Clustering Coefficients				X										B	S			X				
CD: Community Detection			X	X											S			X			X	
GC: Graph Contraction				X									B	B							X	
GP: Graph Partitioning				X							B/S			B							X	
GTC: Global Triangle Counting					X							B							X			
Insert/Delete						X									S		X					
Jaccard				X			B/S															X
MIS: Maximally Independent Set										B				B								
PR: PageRank			X										B					X				
SSSP: Single Source Shortest Path		X							B				B/S	B				X			X	
APSP: All pairs Shortest Path		X												B								X
SI: General Subgraph Isomorphism					X						B/S											
TL: Triangle Listing					X						B/S											X
Geo & Temporal Correlation						X										B/S				X		

Kernel Class: what class of computing kernel performs

Benchmarking Efforts
- S => Streaming
- B => Batch
- B/S => Both

Outputs: what is size or structure of result of kernel execution?

A Real World App

Real World vs. Benchmarks

- Processing more than single kernel
- Many different classes of vertices
- Many different classes of edges
- Vertices may have 1000's of properties
- Edges may have timestamps
- Both batch & streaming are integrated
  - Batch to clean/process existing data sets, add properties
  - Streaming (today) to query graph
  - Streaming (tomorrow) to update graph in real-time
- "Neighborhoods" more important than full graph connectivity

Sample Real-World Batch Analytic
(From Lexis Nexis)

Auto Insurance Co: "Tell me about giving auto policy to Jane Doe" in < 0.1sec

- **2012: 40+ TB of Raw Data**
- **Periodically clean up & combine to 4-7 TB**
- **Weekly "Boil the Ocean" to precompute answers to all standard queries**
  - **Does X have financial difficulties?**
  - **Does X have legal problems?**
  - **Has X had significant driving problems?**

Relationships

  - **Who has shared addresses with X?**
  - **Who has shared property ownership with X?**

Look up answers to precomputed queries for "Jane Doe", and combine

"Jane Doe has no indicators
But
she has shared multiple addresses with Joe Scofflaw
Who has the following negative indicators …."

Sample Analytic Details

- Given: 14.2+ billion records from
  - 800+ million ***entities*** (people, businesses)
  - 100+ million ***addresses***
  - records on who has resided at what address

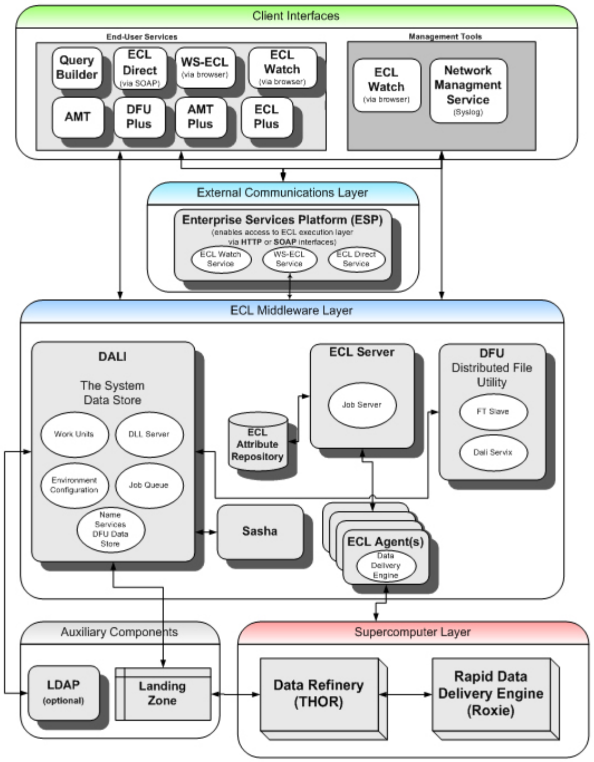
Vertices

Edges

- Goal: for *each* entity ID, find *all other IDs* such that
  - Share at least 2 addresses in common
  - Or have one address in common and "close" last name
  - Matching last names requires processing to check for typos ("Levenshtein distance")
- Akin to a join based on common address, with grouping and thresholding on # of join results
- Dozens of similar analytics computed once a week on 400 node cluster

Sample Batch Implementation Platform: Lexis Nexis

- Entity data kept in huge persistent tables
  - Often with **1,000s** of columns
- Programming in declarative ECL
- **THOR**: runs "offline" on 400+ node systems
  - Batch analytic processing over large data sets
  - Large distributed parallel file system
  - Leaves all data sets for queries in indexed files
- **ROXIE**: runs "online" on smaller system
  - User queries using output files from THOR
  - Dynamically interrogate indexed files
  - Can perform localized ECL on data subsets
- **No dynamic data updates**

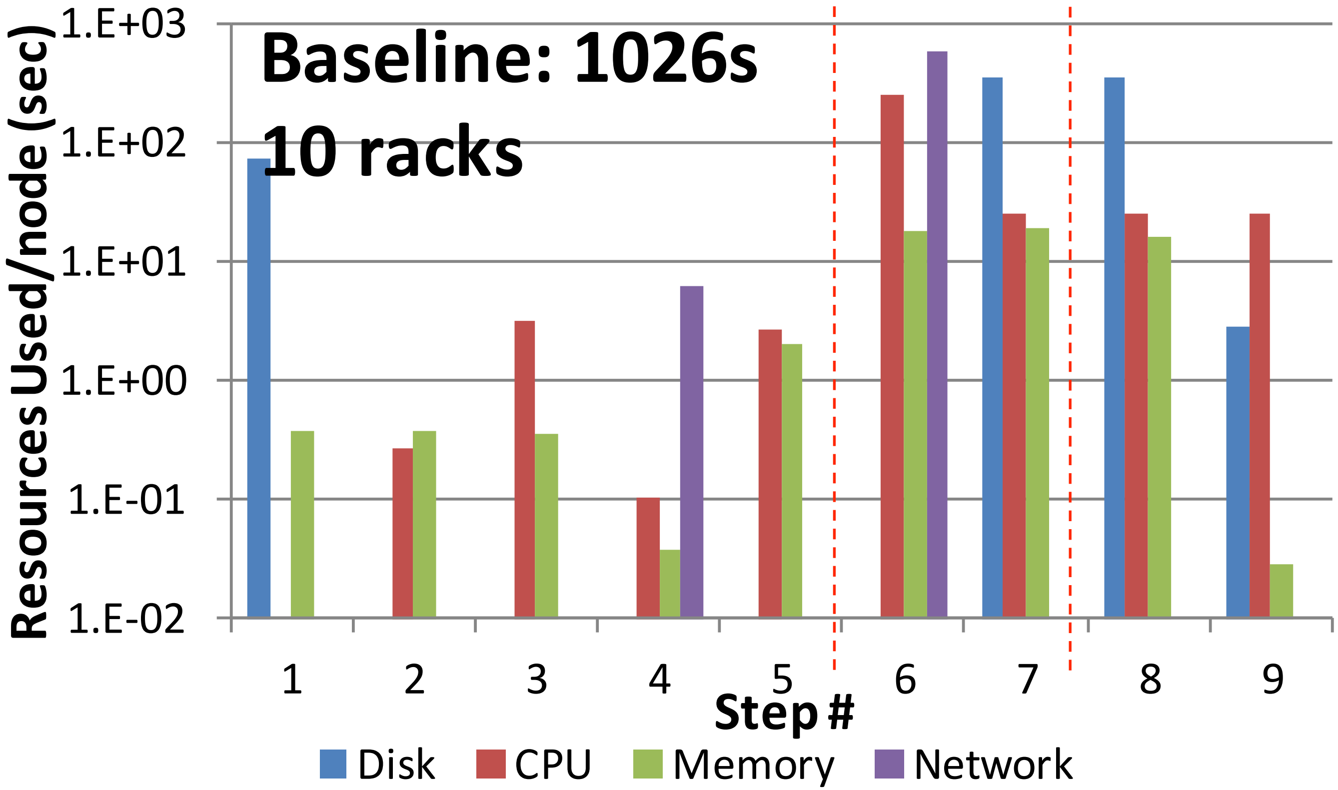


Software Architecture:
https://upload.wikimedia.org/wikipedia/commons/0/02/Fig4b_HPCC.jpg

Execution on Today's Architectures

- Model built to estimate usage of following
  - Bandwidth: Network, Disk, Memory
  - Processing capability
- Baseline: cluster of 400 dual-Xeon nodes
- Menu of improvement options investigated
- "Conventional" improvements
  - No one option >45% increase in performance
  - Significant gains only when all applied at once
- "Unconventional" improvements even better
  - ARMs for Xeons
  - 2-level memory
  - Computing in "3D memory"

A Model Based on Contemporary Architecture



- Optimal code streams data thru multiple kernels till barrier
- No one resource is consistent bottleneck
- Inter-node comm: dynamically random small message

The Core of This Computation as a Benchmark Kernel

Sample Analytic Details

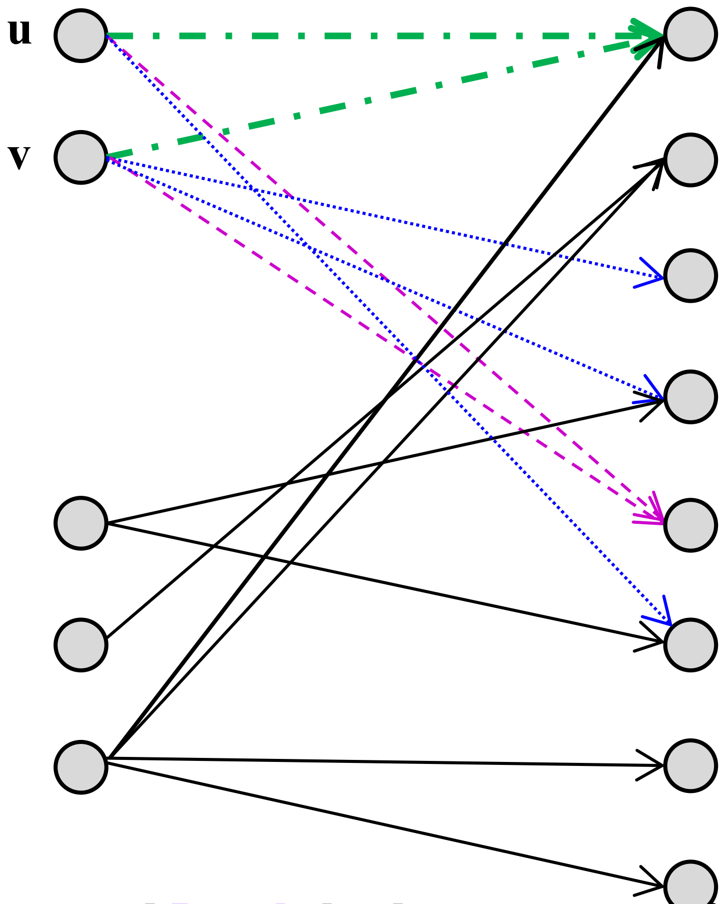
- Given: 14.2+ billion records from
  - 800+ million ***entities*** (people, businesses)
  - 100+ million ***addresses***
  - records on who has resided at what address

Vertices

Edges

- Goal: for *each* entity ID, find *all other IDs* such that
  - Share at least 2 addresses in common
  - Or have one address in common and "close" last name
  - Matching last names requires processing to check for typos ("Levenshtein distance")
- Akin to a join based on common address, but with grouping and thresholding on # of join results
- Dozens of similar analytics computed once a week on 400 node cluster

Neighborhoods & Jaccard Coefficients: The Essence of NORA problems



N(u) = set of neighbors of u

$\Gamma(u,v)$ = fraction of neighbors of u and v that are in common

$\Gamma(u,v) = |N(u) \cap N(v)|/(N(u) \cup N(v)|$

Alternative:

d(u) = # of neighbors of u

$\gamma(u, v)$ = # of common neighbors

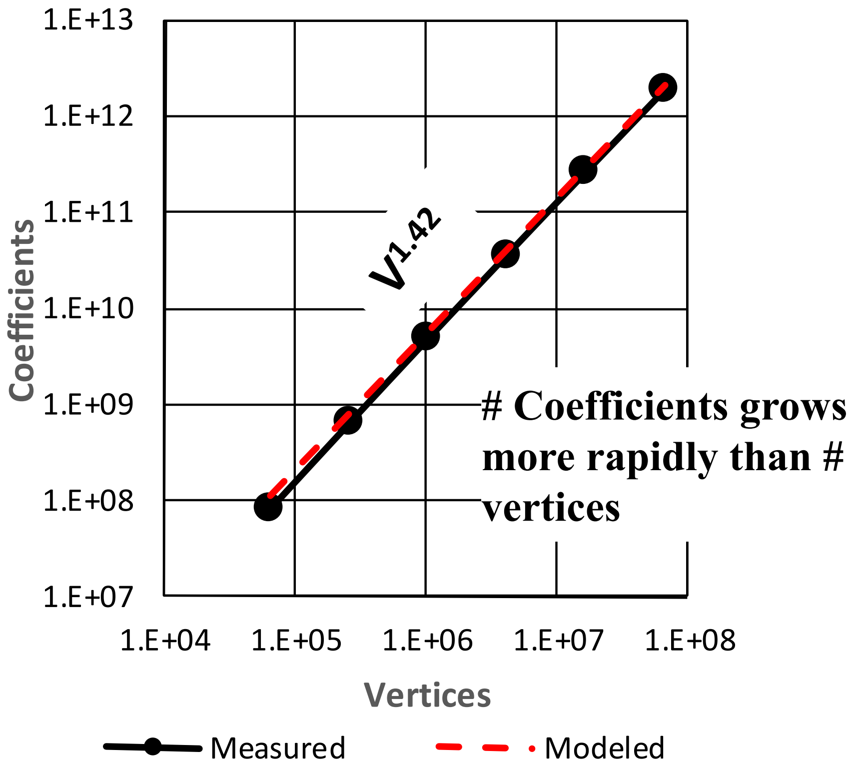
$\Gamma(u,v) = \gamma(u, v) /(d(u)+d(v)- \gamma(u, v))$

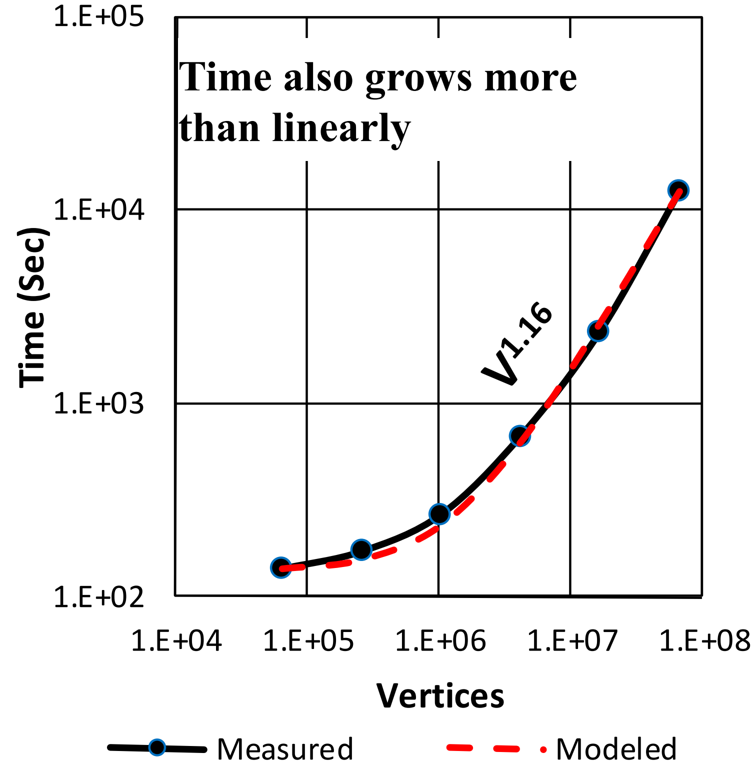
The LexisNexis shared address NORA problem is an extension of this

Green and Purple lead to common neighbors
Blue lead to non-common neighbors

Results of a Map-Reduce Batch Implementation

RMAT matrices, average d(i) = 16, on 1000 node system, each with 12 cores & 64GB





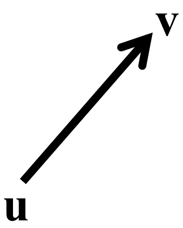
JACS (Jaccard Coefficients / Sec) = 1.6E6*$V^{0.26}$

Entire LN Analytic **approx 10X faster**

Burkhardt "Asking Hard Graph Questions," Beyond Watson Workshop, Feb. 2014.

A Streaming Form: Better Match to Future *Real-Time* NORA

- Assume Edges arrive in stream {(u,v)}
  - Graph keeps just "largest Γ" for each vertex

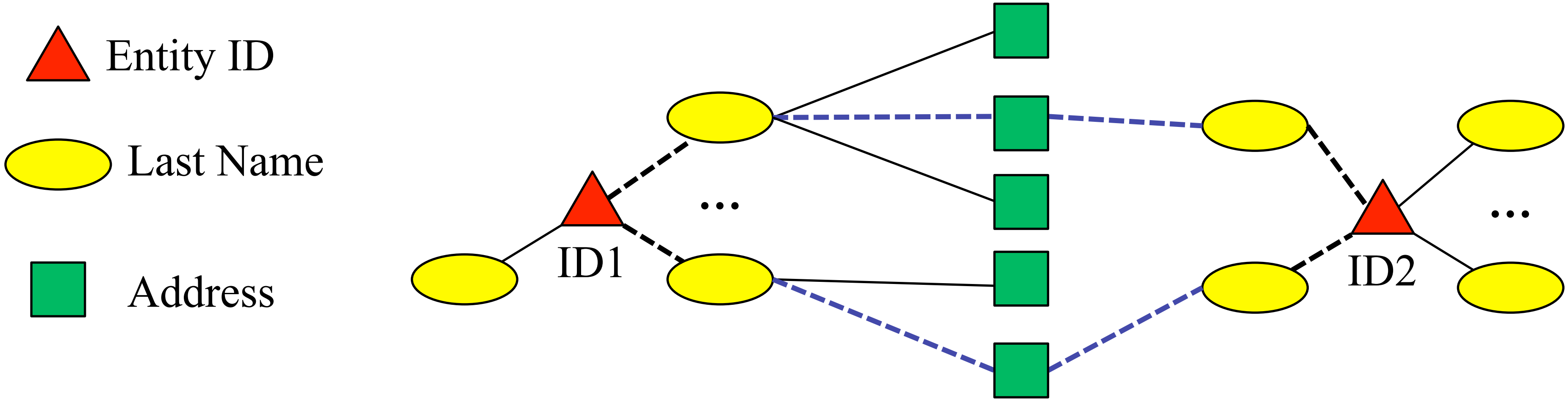


- Question: does any individual edge addition significantly change any vertex's largest Γ
  - Especially to change to cross some threshold
- Implementation involves looking at
  - Neighbors of neighbors of u (to update u's max Γ)
  - Neighbors of v (to update their peak Γ, given u now shares neighbors)
- Bloom filter-like heuristic can bound thresholds early
- Lots of interesting variations

P. Kogge, "Jaccard Coefficients as a Potential Graph Benchmark," GABB, IPDPS, 2016

Looking Forward: Converting Such Problem into Large Graphs

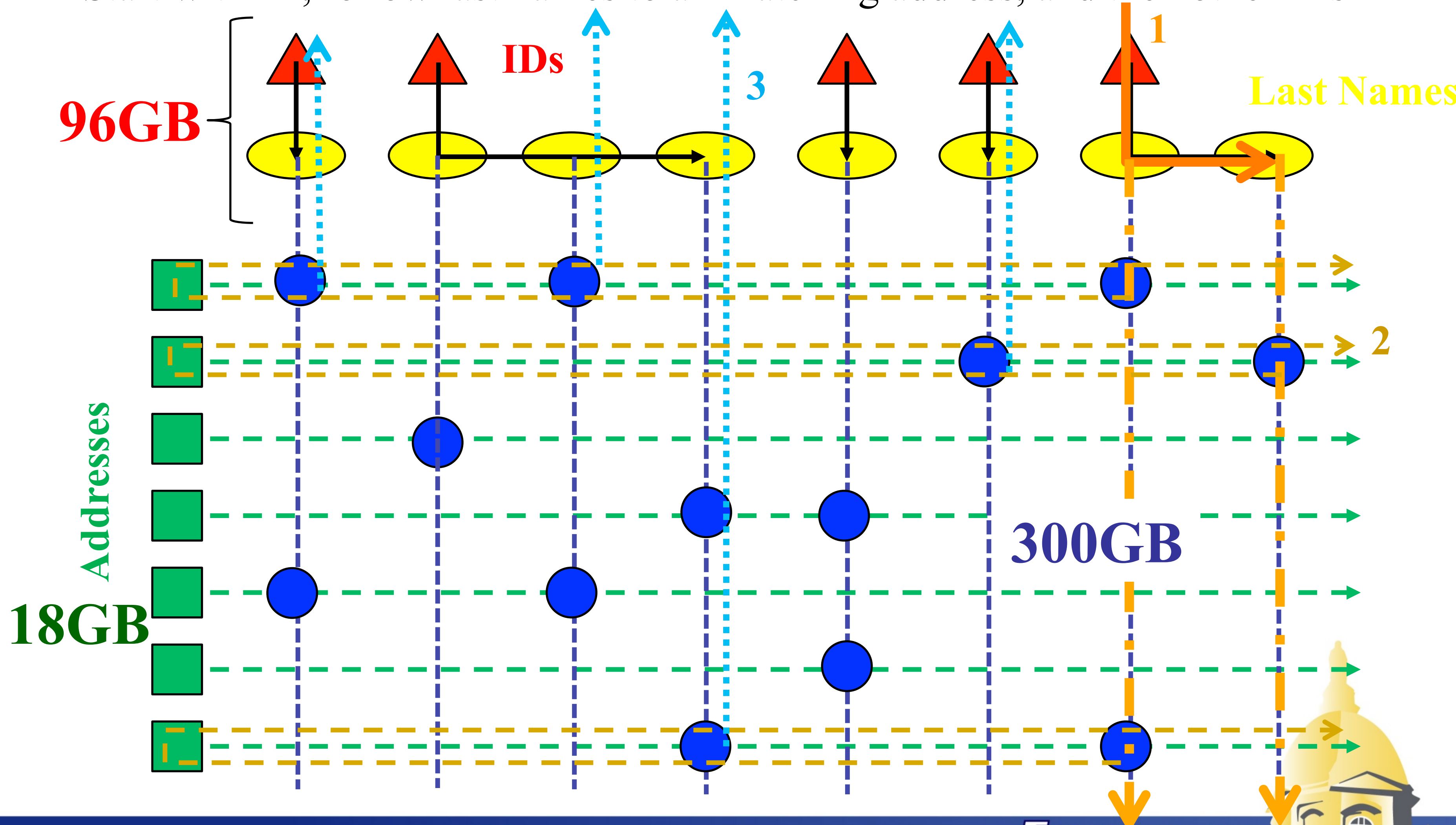
- Create **graph** of name/address records



- Query: Given a specific ID1 find all ID2s that meet requirements
  - Start with ID1 and find all ID2 reachable via shared address
  - Score each path
  - Sum all path scores & pass (ID1, ID2) if > threshold

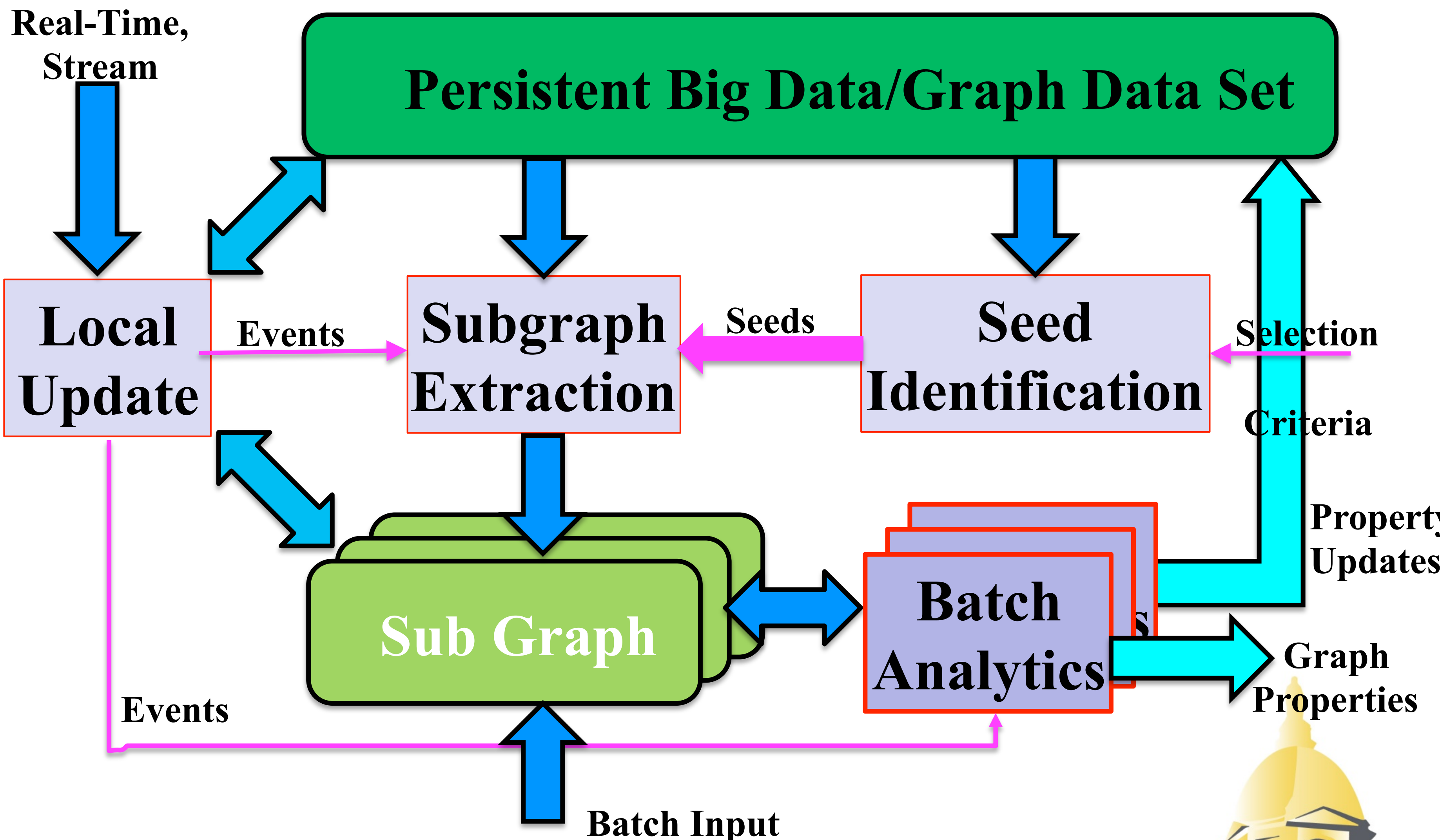
Ability to do *On-Demand* Graph Queries Will Change Business Model

Start with ID, follow last names to all matching address, and then other IDs



Canonical Graph Processing

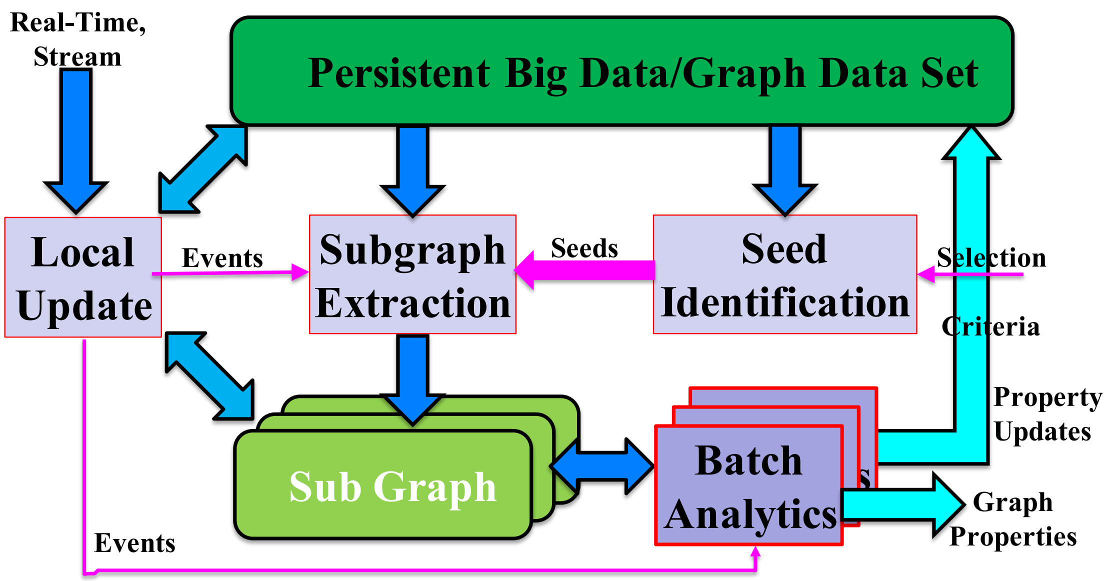
Canonical Graph Processing



Streaming Characteristics

- Two kinds
  - Streams of queries
  - Updates to persistent data
- Both typically localized to start with
- Streaming updates often multi-step
  - Perform update (use atomics)
  - Perform some local computations
  - Compare to threshold
  - If threshold passed, extract some larger subset
  - And perform a bigger analytic

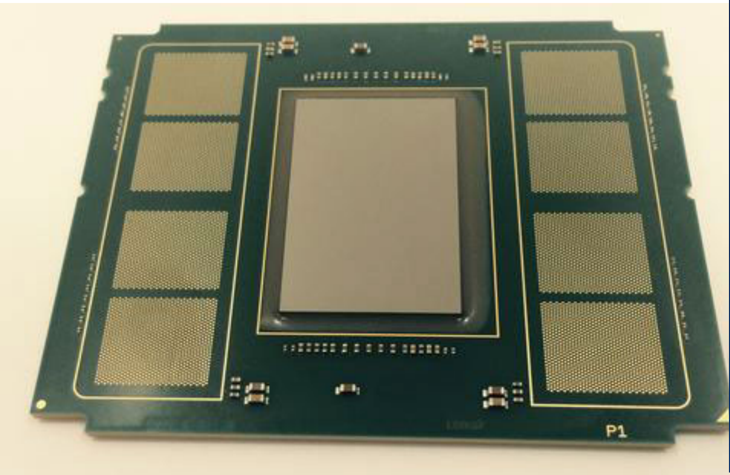
Observations

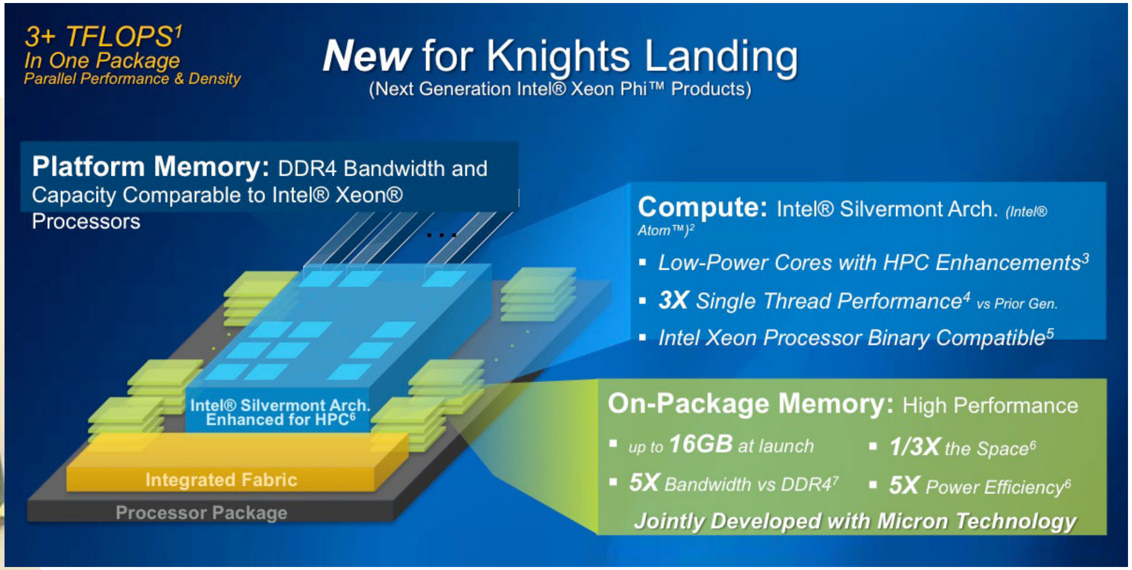


- Data sets live in persistent memory
- Streaming updates trigger threshold tests
- Streaming queries result in local graph traversals
- Batch analytics used primarily for analysis & new property computation

Emerging Architectures to Accelerate Graph Processing

Knight's Landing: 2-Level Memory



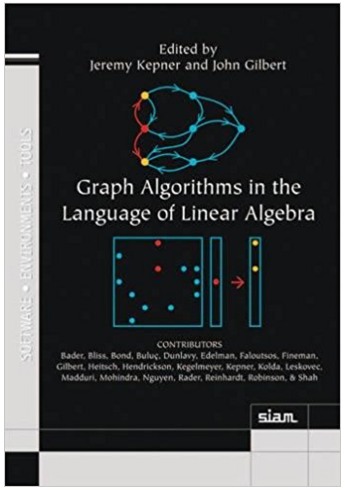


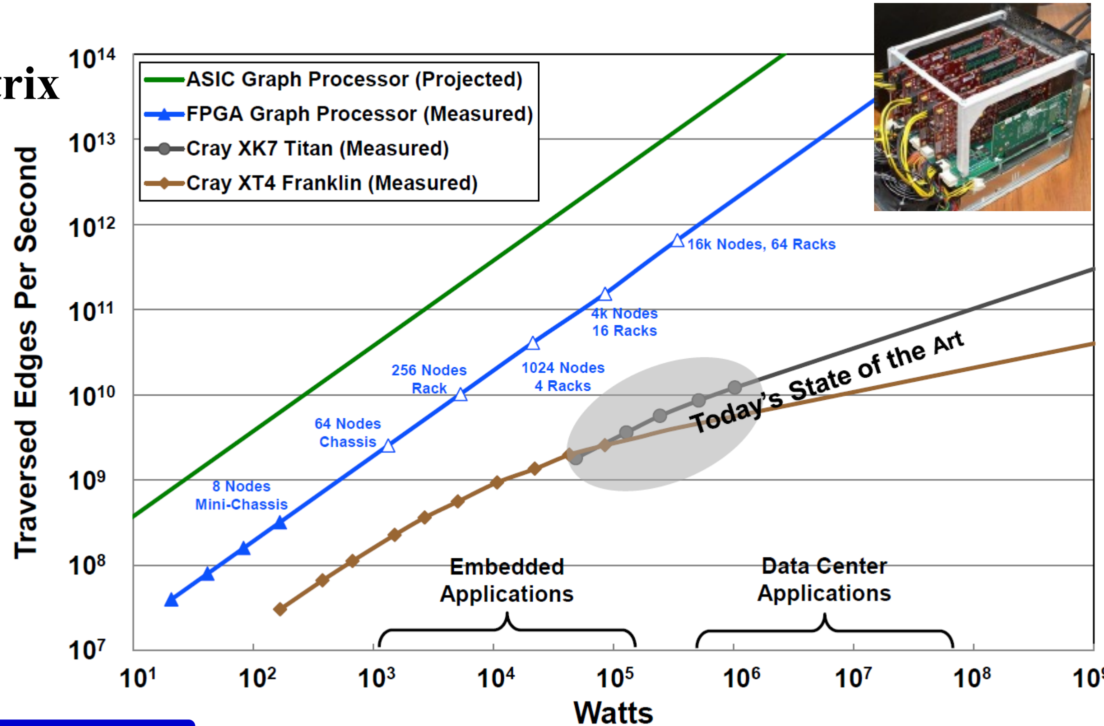
https://www.micron.com/products/hybrid-memory-cube/~/media/track-2-images/content-images/content_image_knights_landing_1200_x_600.jpg?la=en

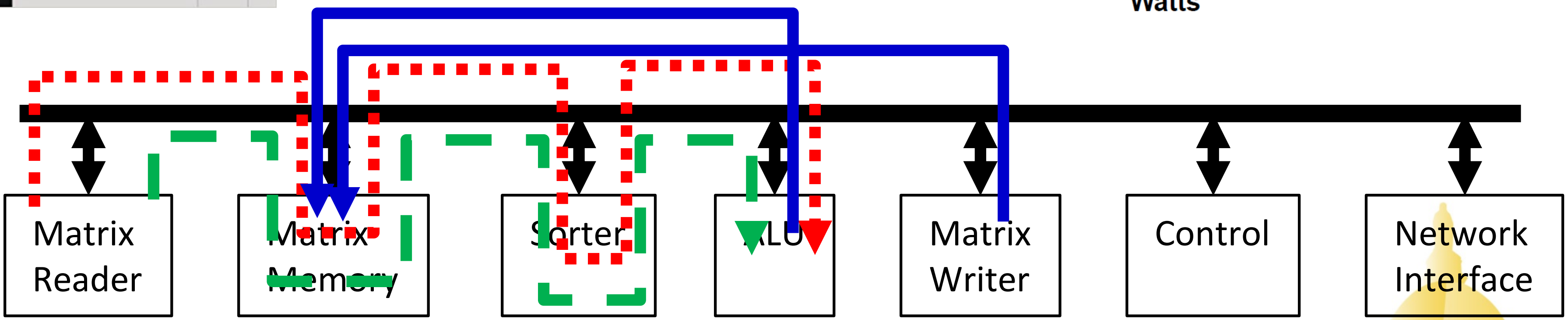
- 2-level memory
  - High capacity for "Persistent Graphs"
  - HBM memory for "Subgraph" Working memory
  - Lots of multi-threaded cores that can see all memory

A Novel Sparse Graph Processor

- Express Graph as Adjacency Matrix
- Graph ops as Matrix-Vector or Matrix-Matrix products







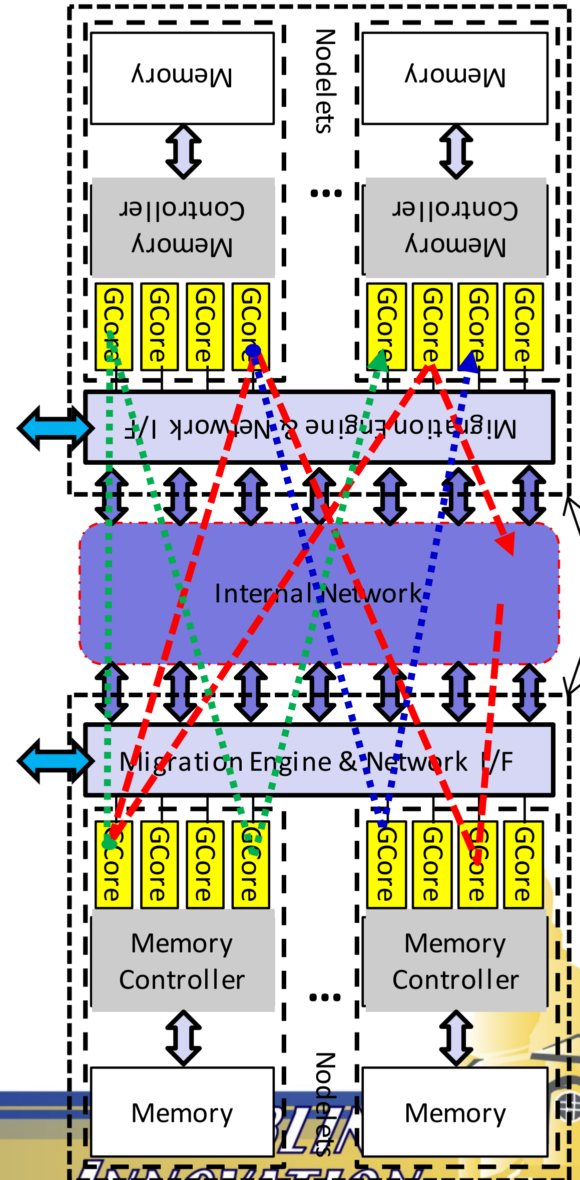
Song, et al. "Novel Graph Processor Architecture, Prototype System, and Results," IEEE HPEC 2016

Migrating Threads for Streaming

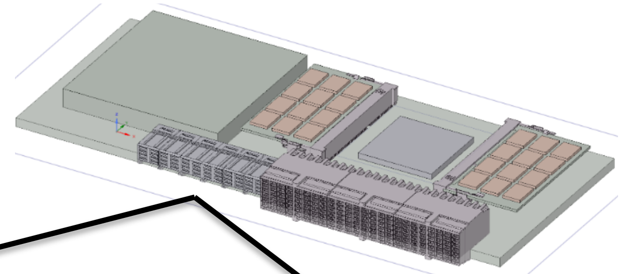
Emu

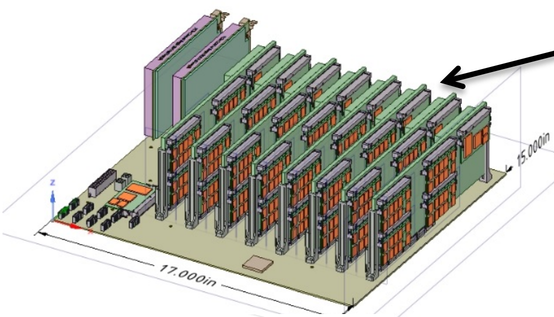
If the mountain won't come to you, you must go to the mountain - ancient proverb

- **Single System-wide Address Space**
- **Parallelism via Memory Channels**
- **Gossamer Cores** (*GCs*) execute **Gossamer Threads** at ***Nodelets***
  - Perform local computations & memory references (inc. atomics)
  - **Migrate to other Nodelets w'o software involvement**
  - **Spawn new Threads**
  - Call System Services on SCs
- **Stationary Cores** *(SCs)*: (Conventional cores)
  - Execute Operating System
  - Manage IO / File System
  - Call or Spawn **Gossamer Threads**
- **Programmed in Cilk**

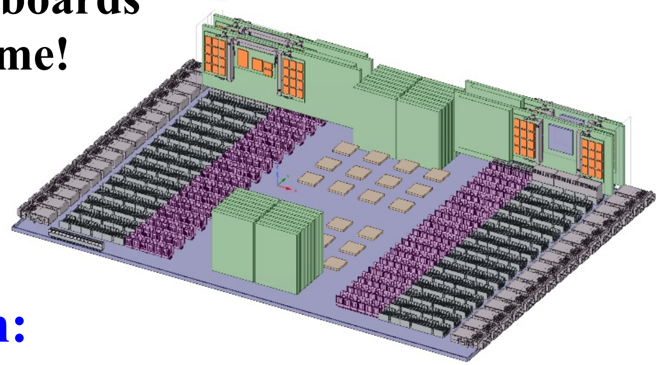


Emu 1





The node boards are the same!



Tool Chain:
Cilk C Today
C++ in progress

Emu Chick

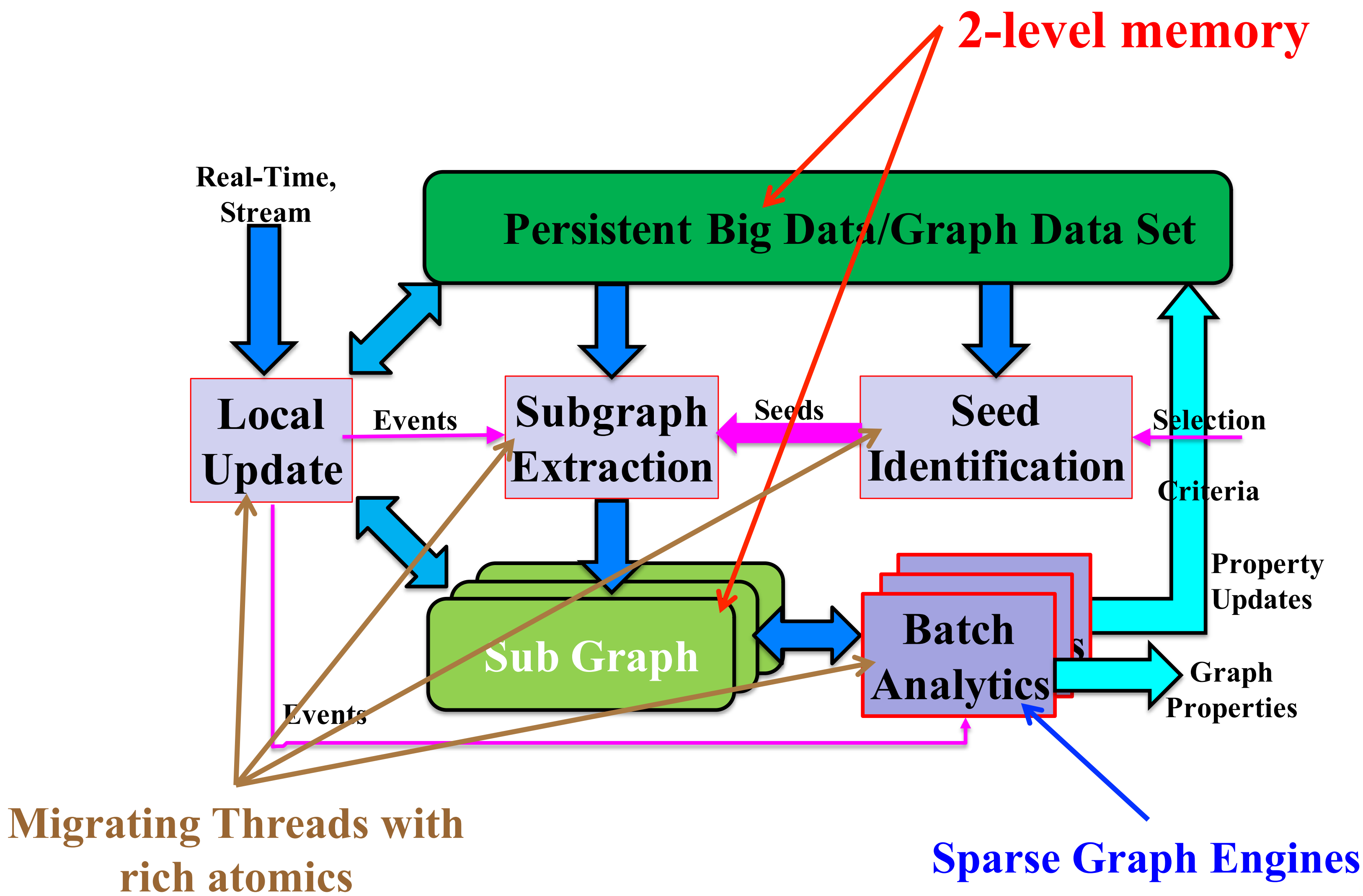
- 8 Nodes; 64 nodelets
- Copy room environment

Emu1 Memory Server

- 256 nodes; 2048 nodelets
- Server room environment
- FCS 2017

T. Dysart, et al. "Highly Scalable Near Memory Processing with Migrating Threads on the Emu System Architecture," SC16

Where Useful



Projection for LexisNexis Problem - Still "Batch Mode" Computations

Emu1 assumes 400MHz GCs
2400 MT/s DRAM Channels

Conclusions

- Most graph benchmarks today
  - Batch oriented
  - Assume simple graphs
  - Focus on total graph properties
- Real-world apps today radically different
  - Many vertex/edge classes with many properties
  - Real interest: localized "neighborhoods"
- Key queries not computable in real-time today
- "Two-level" graph processing flow meshes both streaming and batch computations
- Architectures are emerging that will support such graph processing directly

Acknowledgements: Funding

- In part by NSF CCF-1642280
- In part by the Univ. of Notre Dame