Hierarchical Identify Verify Exploit (HIVE) Program

Trung Tran

DARPA/MTO

Proposers Day Brief

DARPA-BAA-16-52

Distribution A. Approved for public release: distribution unlimited.

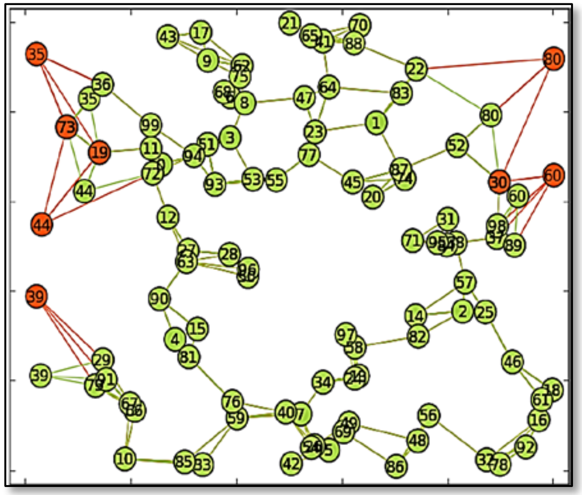
HIVE – What are we trying to do?

HIVE will create a graph analytics processor that achieves 1000x improvement in processing efficiency

- This will enable relationships between events to be discovered as they unfold in the field rather than relying on forensic analysis in data centers
- This will enable data scientists to make associations previously thought impractical due to the amount of processing required

There is an increasing need to understand relationships

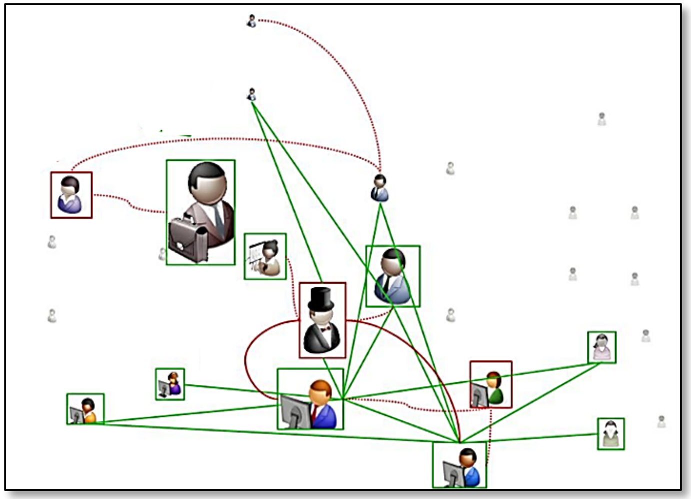
Cyber Security



Which cyber events are probes on the network?

- Who are they probing and who have they infected in the network?
- Only a small number of events are probes – graph is sparse.

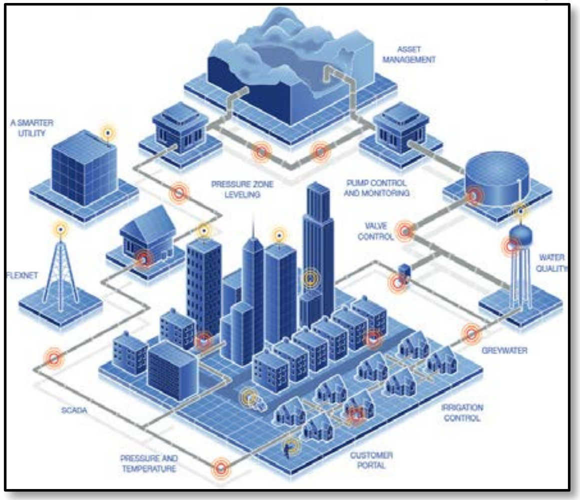
Social Media Analysis



Who influences me to buy a product?

- Who has access to my social media pages and what are they saying to me?
- Since only a few people have direct influence on me – graph is sparse.

Infrastructure Monitoring



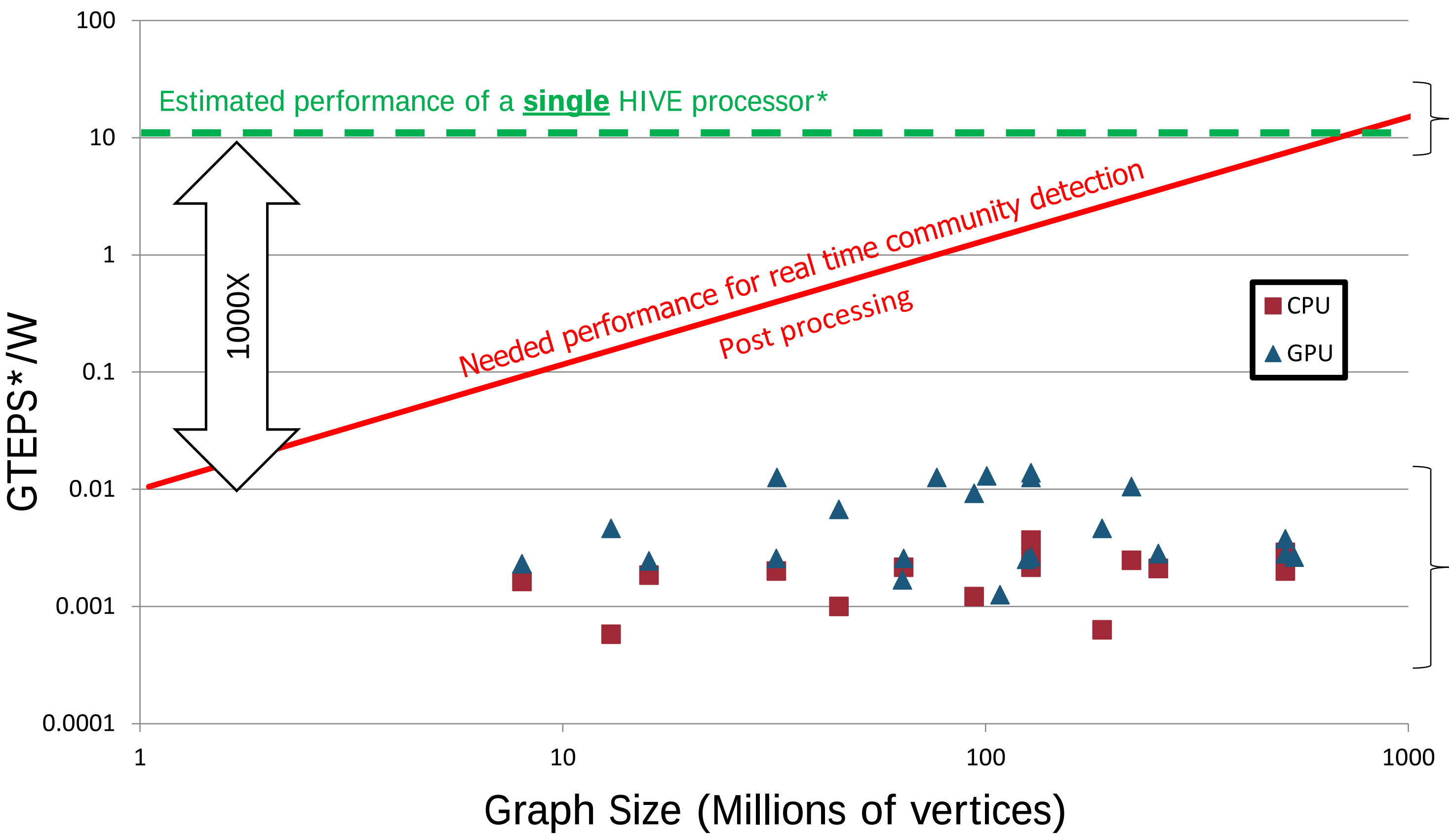
Can I spot failures before they become critical?

- How do I avoid cascading failures and what are the system dependencies?
- Only a small number of critical dependencies – graph is sparse.

Graph analytics is beginning to be applied to a broad set of problems

Distribution A. Approved for public release: distribution unlimited.

Graph analytics today requires large data centers



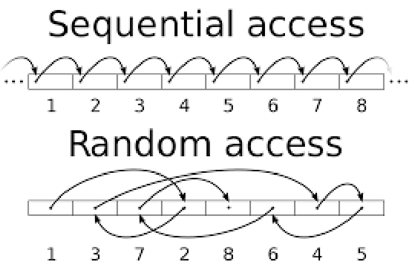
The HIVE program will develop a single processor capable of efficiently performing community detection on a graph of up to a billion vertices in real time.

Current single chip CPU/GPU hardware cannot efficiently process large graphs in real time. To overcome processor limitations, large data centers are required.

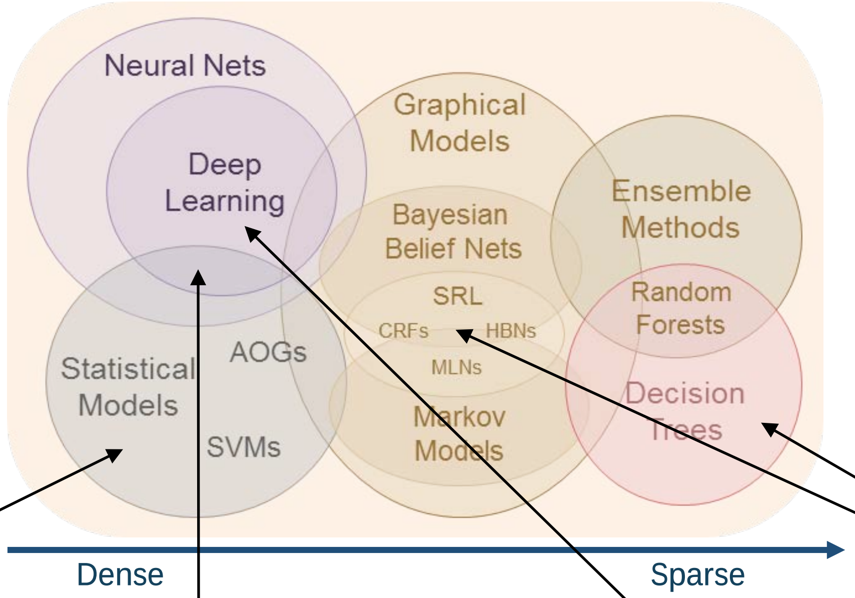
HIVE aims to enable scalable, real-time graph analytics at the network edge

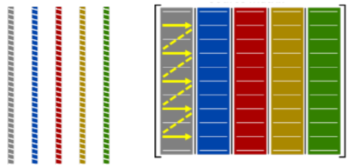
* GTEPS = Giga Traversed Edges Per Second

HIVE – Today’s hardware is focused on dense data



Sequential access is <u>good for dense data</u> Sparse data requires random access





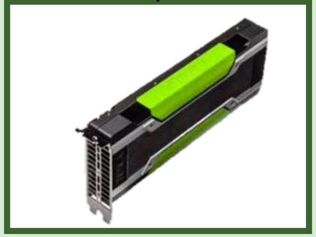
<u>Lower level primitives</u>
(5x5 Matrix)

- 25 Scalar operations
- 5 Vector operations
- 1 Matrix operation

Intel CPU

- Sequential processing
- Sequential memory access
- Slow (20GB/s) to memory
- Limited scalability (16GB/s)
- Optimized for Statistics

Source: Intel



Nvidia GPU

- Parallel processing
- Sequential memory access
- Faster (288GB/s) to memory
- Limited scalability (20GB/s)
- Used for CNNs

Source: Nvidia

Google TPU

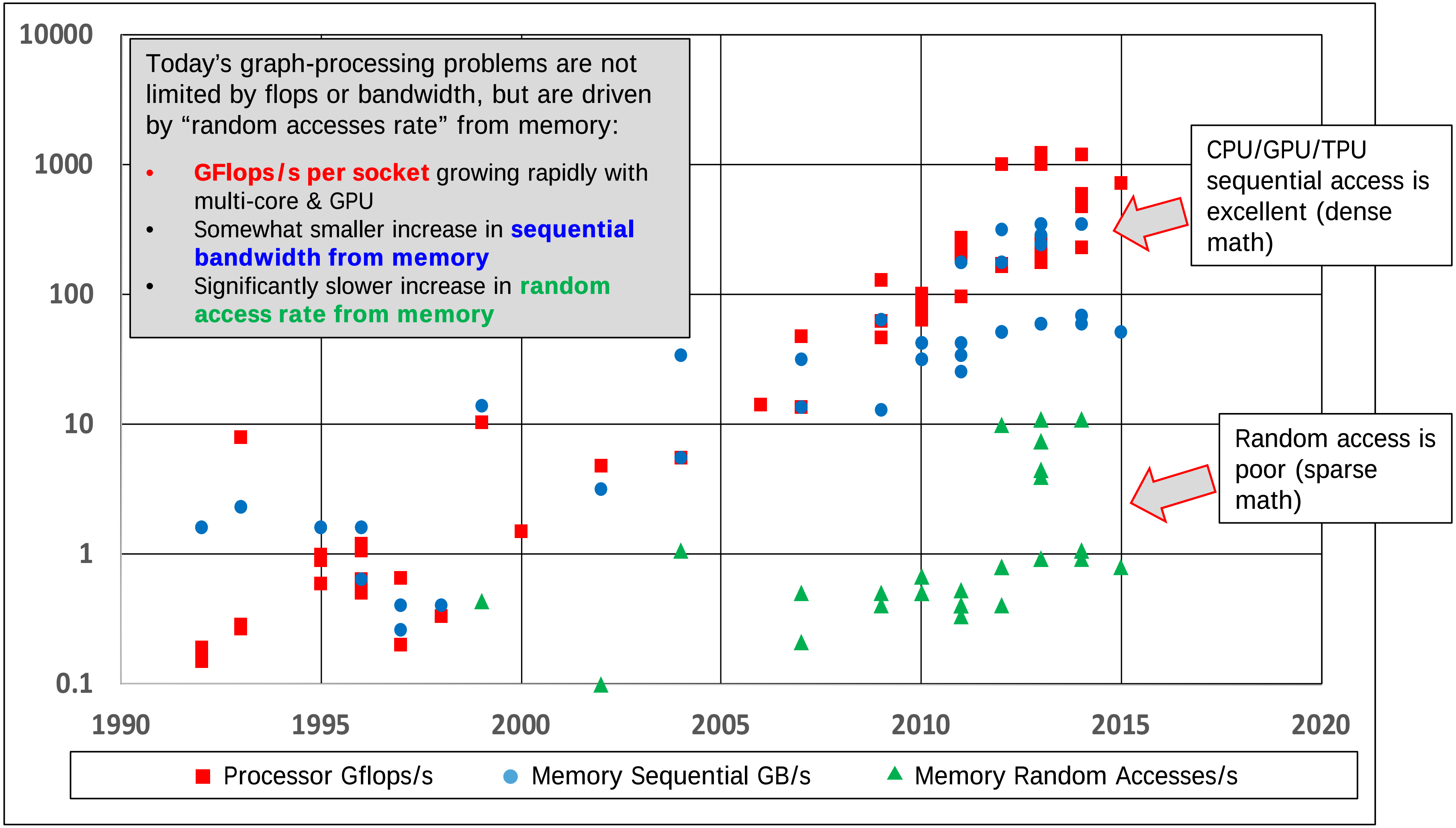
- Parallel processing
- Sequential memory access
- Slow (20GB/s) to memory
- Limited scalability (16GB/s)
- Optimized for DNNs

Source: Google

HIVE

- Parallel processing
- Parallel memory access
- Fastest (TB/s) to memory
- Higher scalability (TB/s)
- Optimized for Graphs

Current GPU/CPU/TPU solutions have poor random access rate



Today’s graph-processing problems are not limited by flops or bandwidth, but are driven by “random accesses rate” from memory:

- **GFlops/s per socket** growing rapidly with multi-core & GPU
- Somewhat smaller increase in **sequential bandwidth from memory**
- Significantly slower increase in **random access rate from memory**

CPU/GPU/TPU sequential access is excellent (dense math)

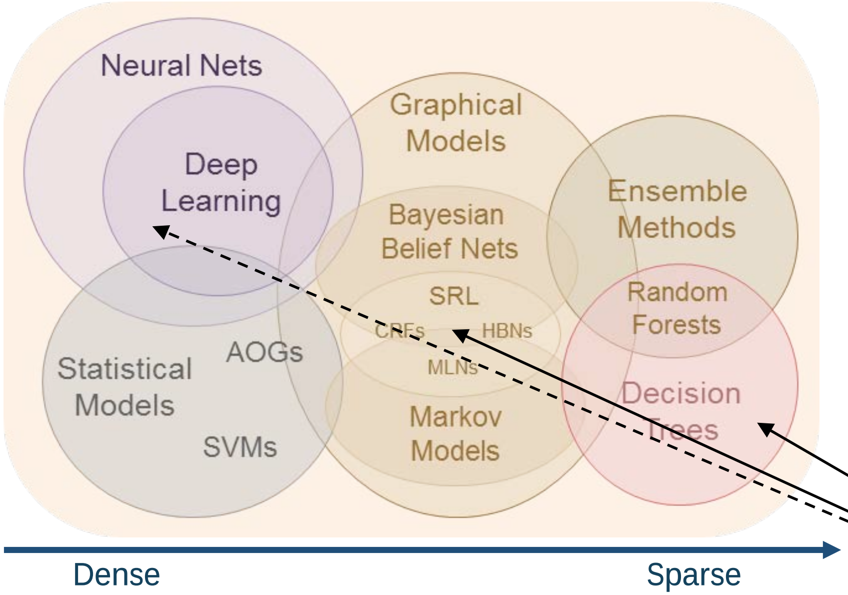
Random access is poor (sparse math)

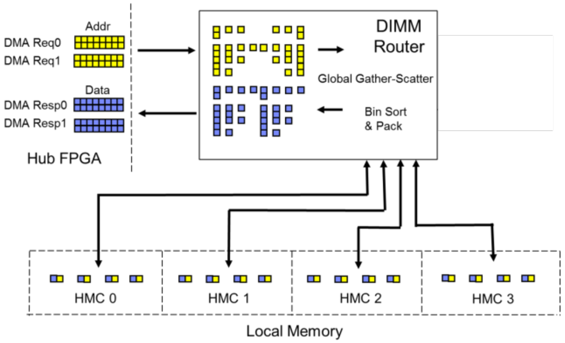
* EMU Technologies Design Review. March 2016

HIVE is focused on sparse data

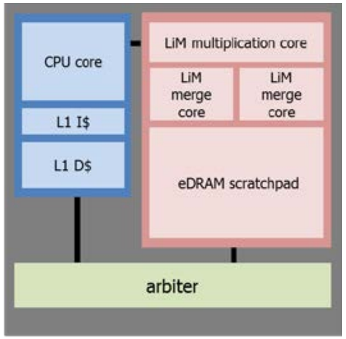
New Approach

Create a new hardware architecture optimized for sparse graphical models and decision trees





Parallel memory access enables random access



Graph Primitives
SpRef/SpAsgn, SpGEMM, SPEWiseX, Scale, Reduce, and Apply

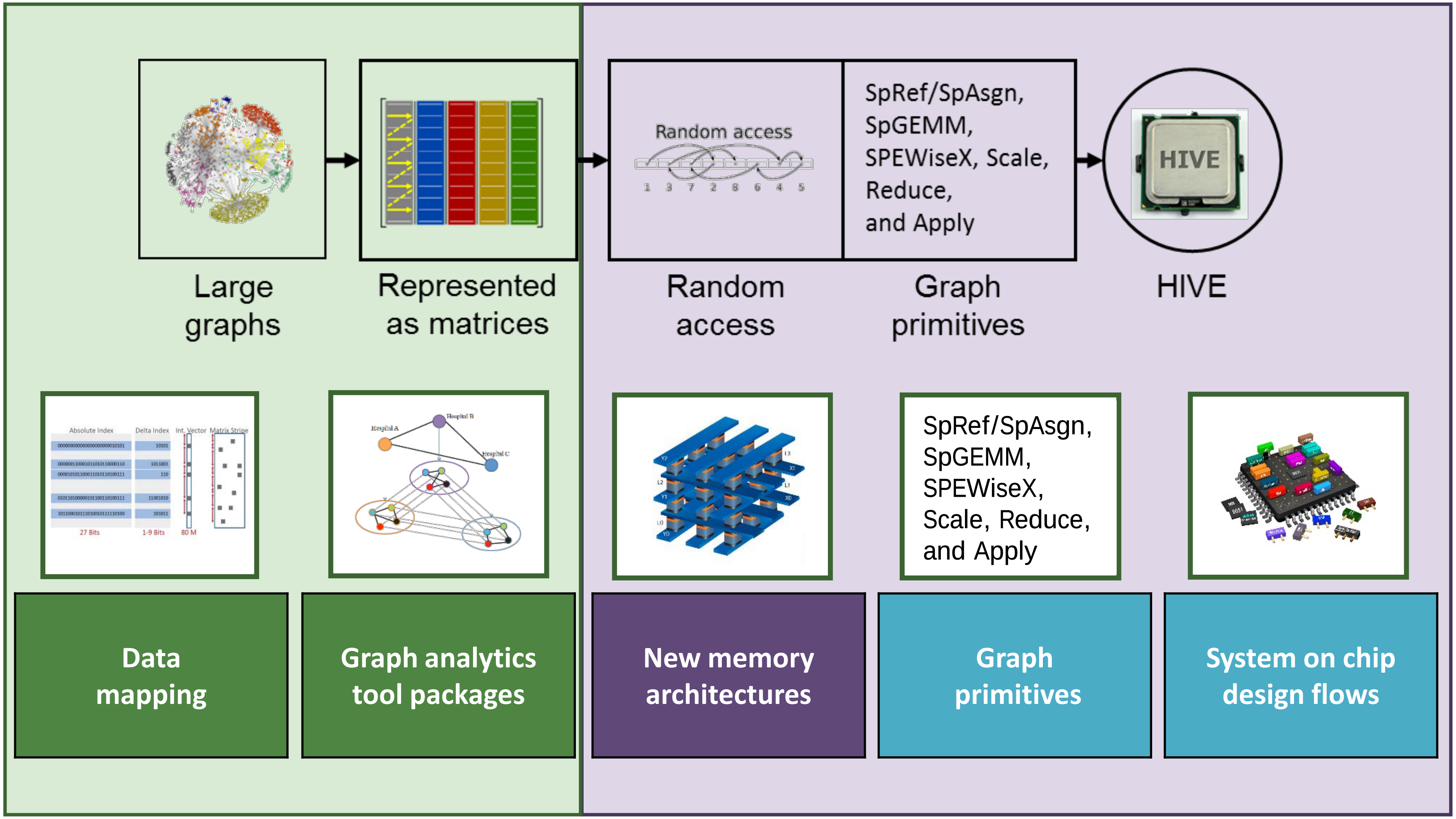
Matrix data format/operations enables more efficient processing

HIVE

- Parallel processing
- Parallel memory access
- Fastest (TB/s) to memory
- Higher scalability (TB/s)
- Optimized for Graphs

Distribution A. Approved for public release: distribution unlimited.

HIVE – Why do we think it will be successful?



Large graphs → Represented as matrices → Random access

SpRef/SpAsgn, SpGEMM, SPEWiseX, Scale, Reduce, and Apply

Graph primitives → HIVE

SpRef/SpAsgn, SpGEMM, SPEWiseX, Scale, Reduce, and Apply

Data mapping

Graph analytics tool packages

New memory architectures

Graph primitives

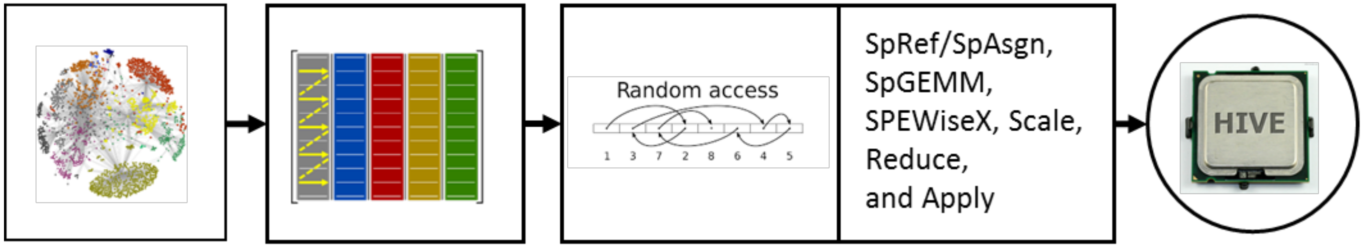
System on chip design flows

Distribution A. Approved for public release: distribution unlimited.

HIVE – Technical challenges

	Define graph primitives	Create data format model	Define data flow model
Graph Software What should be accelerated?	Define linear algebra building blocks which can be accelerated	Map graph matrix into subarrays which can allow for easy memory mapping	Define data movement from processor to memory and between processors
	Accelerators	Memory	Scaling
Graph Accelerator How should it be accelerated?	Develop hardware accelerators for each building block	Create memory controller which optimizes data movement based on mapping	Develop bus architectures to avoid congestion in data movement

HIVE – Program structure



SpRef/SpAsgn, SpGEMM, SPEWiseX, Scale, Reduce, and Apply

Large graphs → Represented as matrices → Random access → Graph primitives → HIVE

- Targeting graph applications
- Dense and sparse data
- Built for scaling
- Intended for embedded

Challenge problem areas

TA 2:
Graph analytics toolkits
(ref: Tensorflow, CUDA)

- Enable real-time streaming graph analytics
- Designed for hardware acceleration
- Generally applicable across a series of graph problems

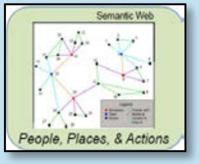
TA 1:
Graph analytics accelerator
(ref: TPU, GPU)

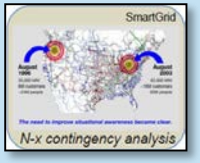
- Runs at <20W
- Enable reduced data movement/processor idle time to <50%
- Allow for 95% memory BW efficiency at 100% random access (R/W)

Evaluation framework

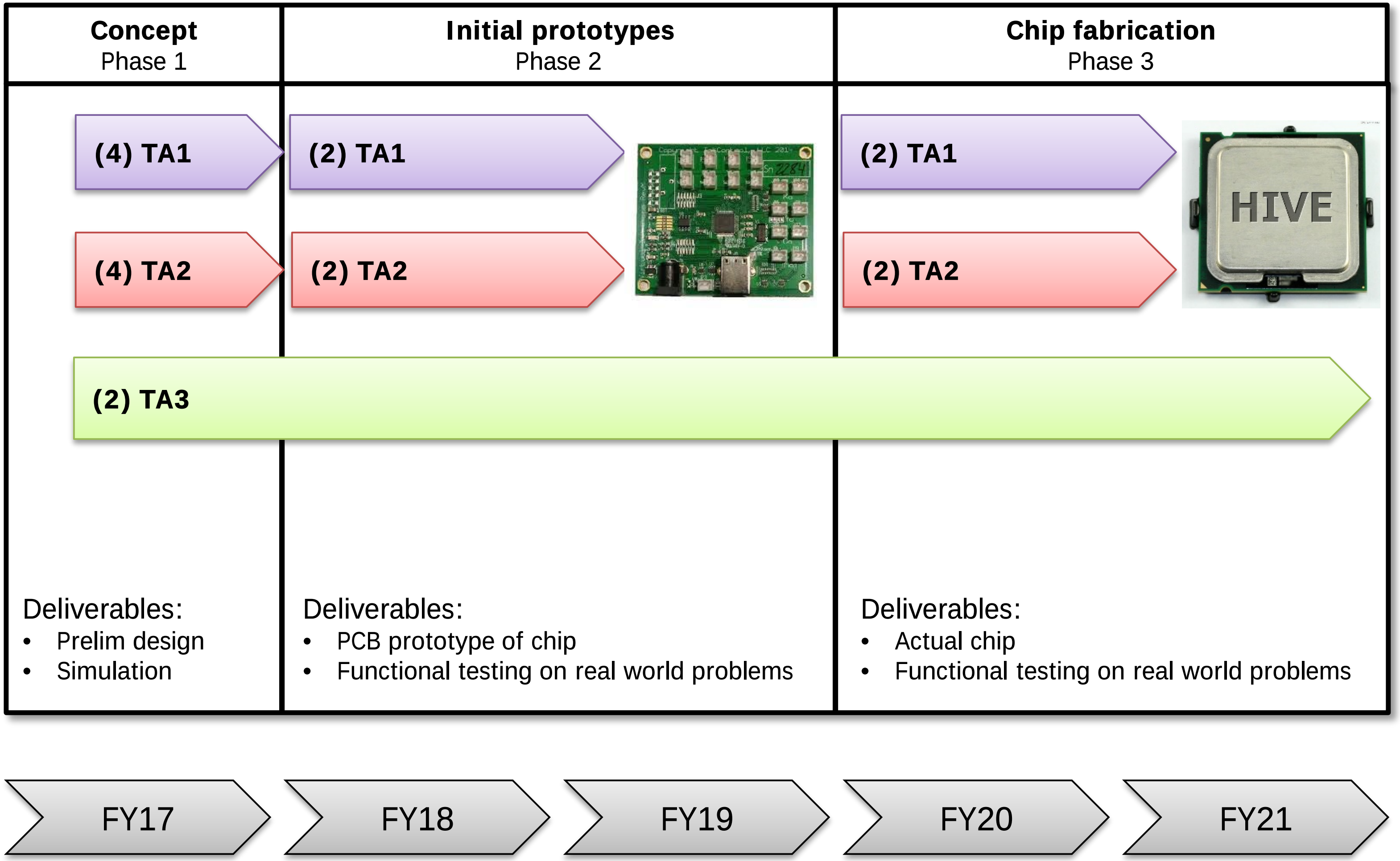
100X improvement

TA3:
Evaluator





HIVE – Program plan



Concept
Phase 1

Initial prototypes
Phase 2

Chip fabrication
Phase 3

Deliverables:

- Prelim design
- Simulation

Deliverables:

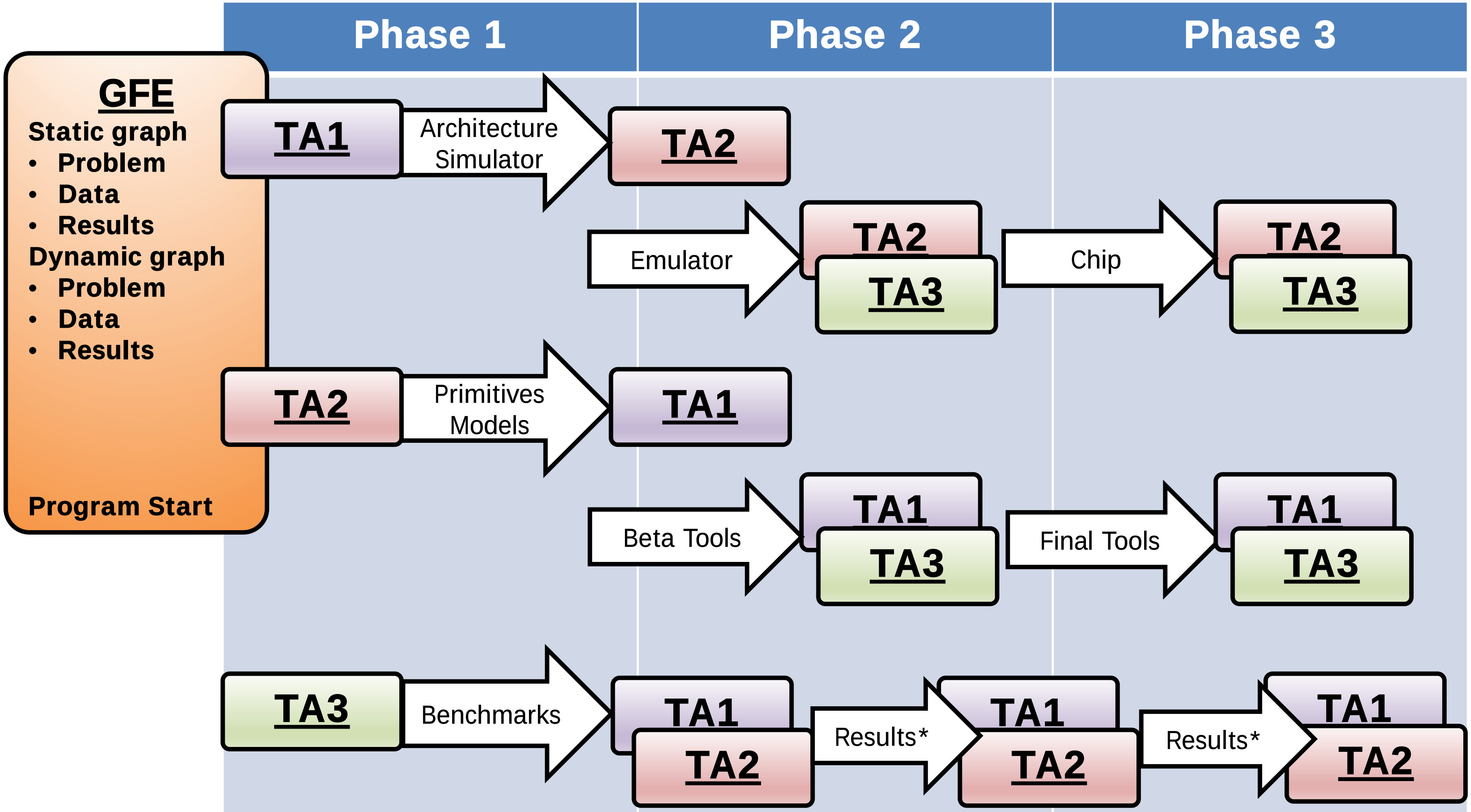
- PCB prototype of chip
- Functional testing on real world problems

Deliverables:

- Actual chip
- Functional testing on real world problems

Distribution A. Approved for public release: distribution unlimited.

HIVE – Dependency table



Phase 1 | Phase 2 | Phase 3

GFE

Static graph
- **Problem**
- **Data**
- **Results**

Dynamic graph
- **Problem**
- **Data**
- **Results**

Program Start

TA1 → Architecture Simulator → TA2

Emulator → TA2 TA3 → Chip → TA2 TA3

TA2 → Primitives Models → TA1

Beta Tools → TA1 TA3 → Final Tools → TA1 TA3

TA3 → Benchmarks → TA1 TA2 → Results* → TA1 TA2 → Results* → TA1 TA2

* Note: Government independent testing may be done on TA3 results

www.darpa.mil

Distribution A. Approved for public release: distribution unlimited.