# UpDown: Breakthrough Graph Accelerator: Streaming, Real-time, Scalable Graph Computing

Andrew A Chien
University of Chicago and Chicago UpDown Computig

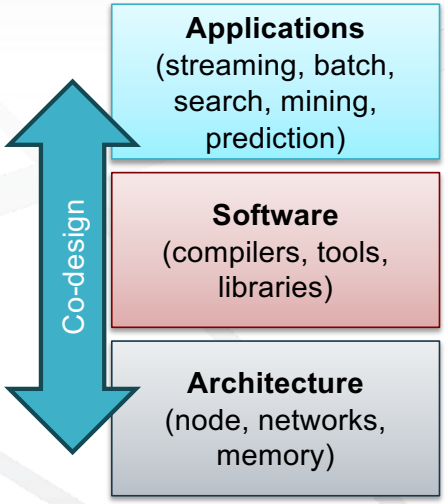
Graph Data Council, TUC Meeting @ VLDB
Boston, MA

September 4, 2026

Chicago UpDown

# Chicago UpDown Computing

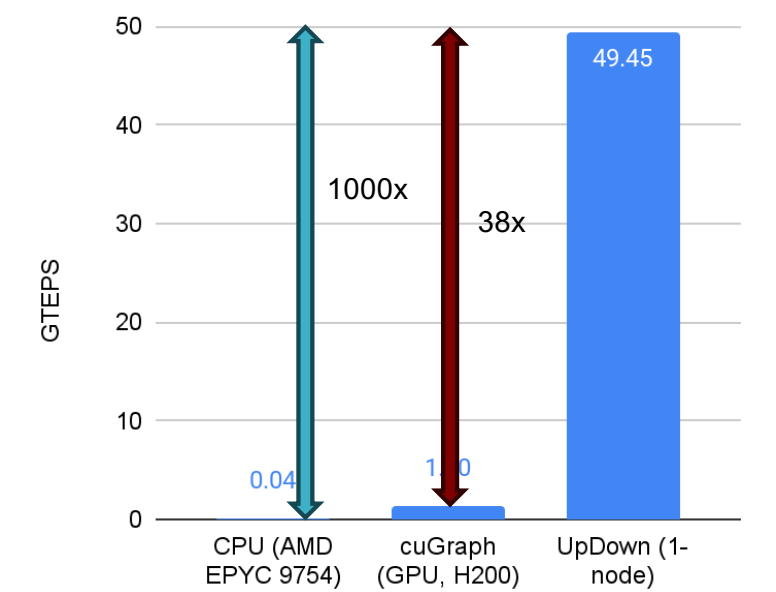
- US Govt, IARPA AGILE Research Program 2022-26
- Deep co-design (applications, software, hardware), produced Graph Computing Acceleration (efficient, scalable, high performance) for sparse
- Product: Complete Software Stack, Hardware Architecture Design.
- & Many application demonstrations




- UpDown Team!

Chicago UpDown

# 1-node UpDown Outperforms CPU's & GPU's




K-truss: Fundamental Graph Algorithm, includes Triangle count + per-edge attribution K=kmax

Similar results for key graph algorithms: Triangle Count, Louvain, Connected Components, Strongly Connected Components, PageRank, BFS, K-cores + more

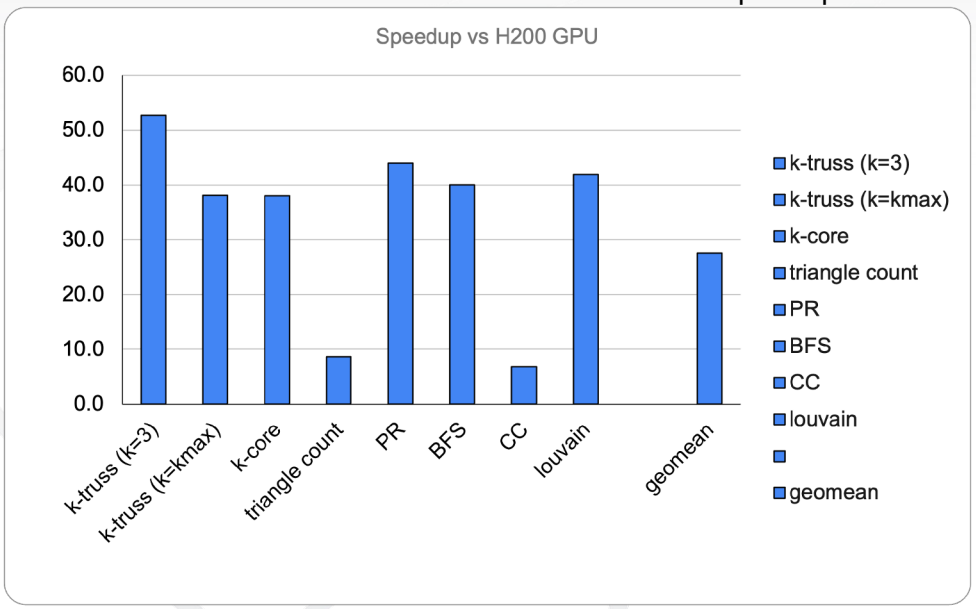
- UpDown Accelerator, winner of MIT Graph Challenge Innovation Award! (2025)
- Cit-Patents: difficult highly-skewed graph of 16.5M edges, STRONG scaling,
- Single node performance is foundation for scalability and efficiency

Chicago UpDown

# Outperforms H200 GPU on Many Graph Benchmarks

- Breadth-first Search
- Pagerank
- Connected Components
- Strongly Connected Components
- K-cores
- K-truss
- Triangle Count
- Louvain Modularity (community detection)
- Also
  - Bioinformatics Assembly
  - Streaming Graph Queries (low latency)
  - Graph Inference and Prediction
- …

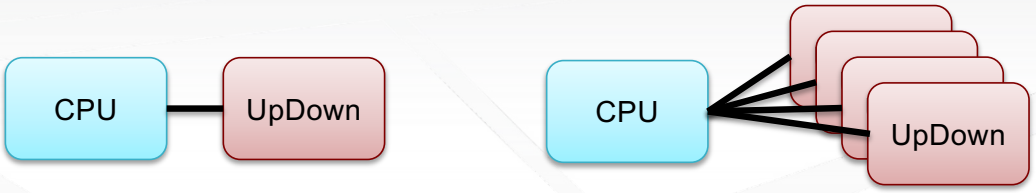
1-node speedups

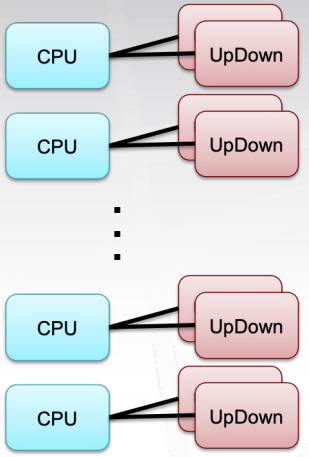



=> at a significant power advantage!

Chicago UpDown

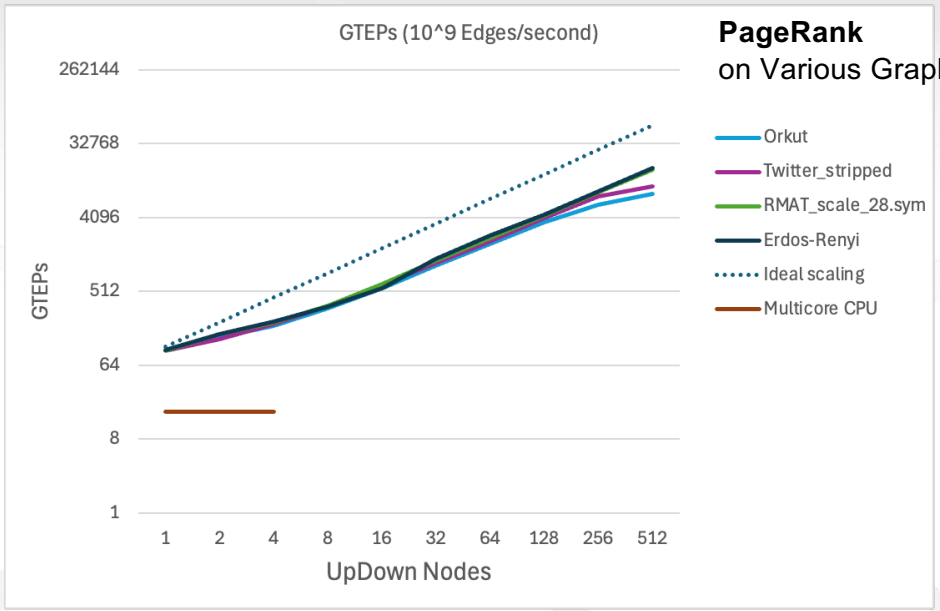
# UpDown Systems scale!




- Traditional accelerator model; scaleup to very large systems
- Key: Simple programming
  - Program runs on host (Python, Julia, Java, C,…)
  - Accelerated kernels on UpDown)
    - simple vertex, edge parallelism
    - ”graph kernels” in event-driven language
    - Manage massive parallelism with Map-reduce
- Same program for all scales! (yes identical)
- IARPA UpDown system designed for 16,384 nodes

# Easy Scaling to Medium (16n) and Large systems (512n)



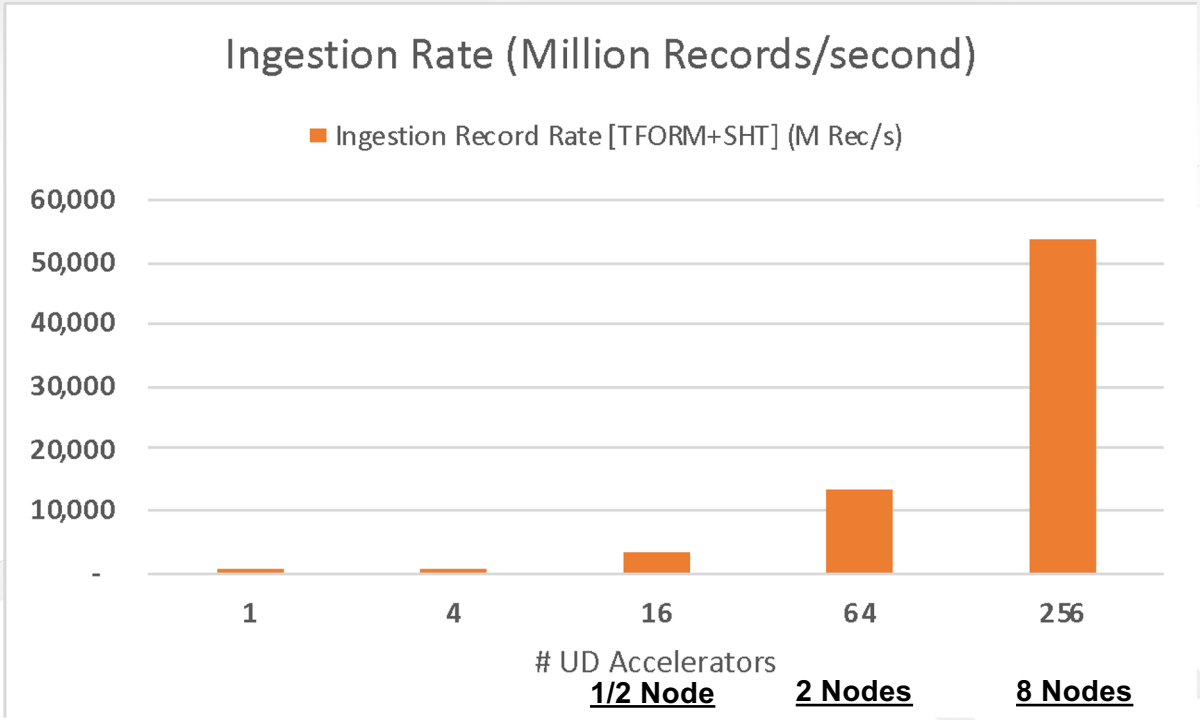

| Algorithm | Speedup | Absolute Performance |
|---|---|---|
| Pagerank | 314,000 | 16,433 GTEPs |
| Connected Components | 216,000 | 9,821 GTEPs |
| Breadth-first Search | 186,000 | 13,960 GTEPs |

And scales on many other graph algorithms…

- Performance exceeds 1,000 times a multicore CPU, and surpasses Supercomputers
  - Easy scaling on irregular graphs is extraordinary!
- Strong Scaling (0.25 B vertex, 4B edge RMAT-28 extreme skew graph); Larger graphs are easier
- UpDown's programmability enables a single program to scale from 1 to 16,384 nodes
- Scaling of graph compute on skewed graphs without graph partitioning [Automatic Management]

Chicago UpDown

# Ingestion Rate and Graph Creation (AGILE 1A, 1B)




- Reading data from CSV, JSON and inserting records into graph data structure
- Ingestion rate is 6.8 billion records/second/node (1 chip).
- 64-128 nodes (rack) => 440-880 Billion records/second
- 4096 Nodes => 27 Trillion records/second
- 16384 Nodes => 108T records/second, Internet (500Tbps,2023)

Chicago UpDown

# How to deliver this for mainstream graph computing community?

- Python, NetworkX, …
- Spark, GraphX, …
- OpenCypher, Gremlin, …

- We have built a set of lower level programming tools to access graph acceleration

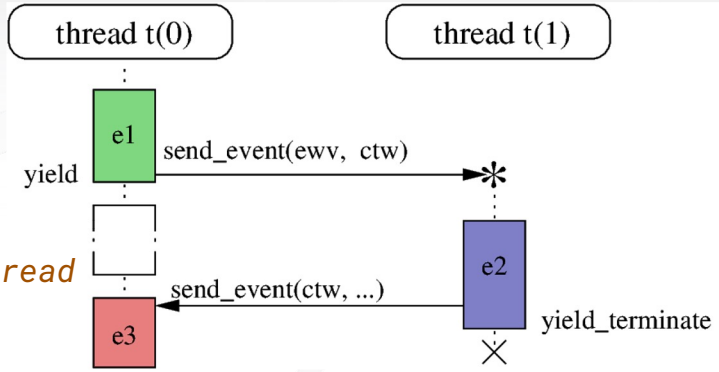
Chicago UpDown

# Programming Graph Kernels

```
thread TCallReturn {
        event e1() {
                print("I am in e1");
                uint64_t evw = evw_new(curNetworkID+1, e2);
                uint64_t ctW = evw_update_event(CEVNT, e3);
                // send uint64 0 and uint64 1 to e2 on a new thread
                send_event(evw, 0, 1, ctW);
                yield;
        }

        event e2(uint64_t data0, uint64_t data1) {
                print("I am in e2 and received this data: %lu, %lu", data0,
data1);
                send_event(CCONT, IGNRCONT);
                yield_terminate;
        }

        event e3() {
                print("I am back from e2");
                //...
        }
}
```




- Event-driven C-like language (Actors, React)
- Thread/memory control

Chicago UpDown

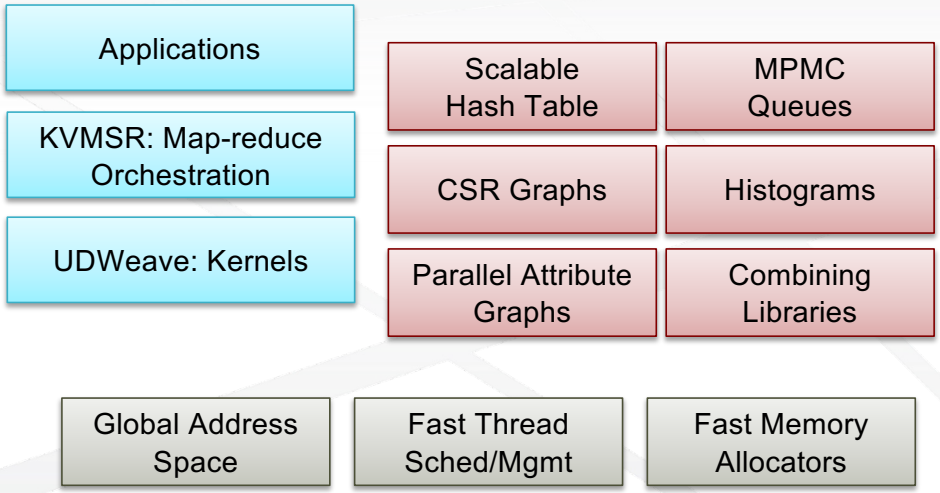
# Large-scale Graph Algorithms

- Kernels: C-like event-driven language (Actors, React)
- Coordinate at-scale with data-parallel loops, map-reduce in a global address space
- Sound familiar?

- "Programming is like multicore programming, but without worrying about memory bandwidth, caches, or thread management."
- Example: BFS, PR, K-truss written in GBBS/Ligra like programming style, auto-load balanced
  - You saw the performance!
- This is the target for graph system implementers (eg. gremlin, cypher, …) for accelerated computation.

Chicago UpDown

# Current Software, UpDown

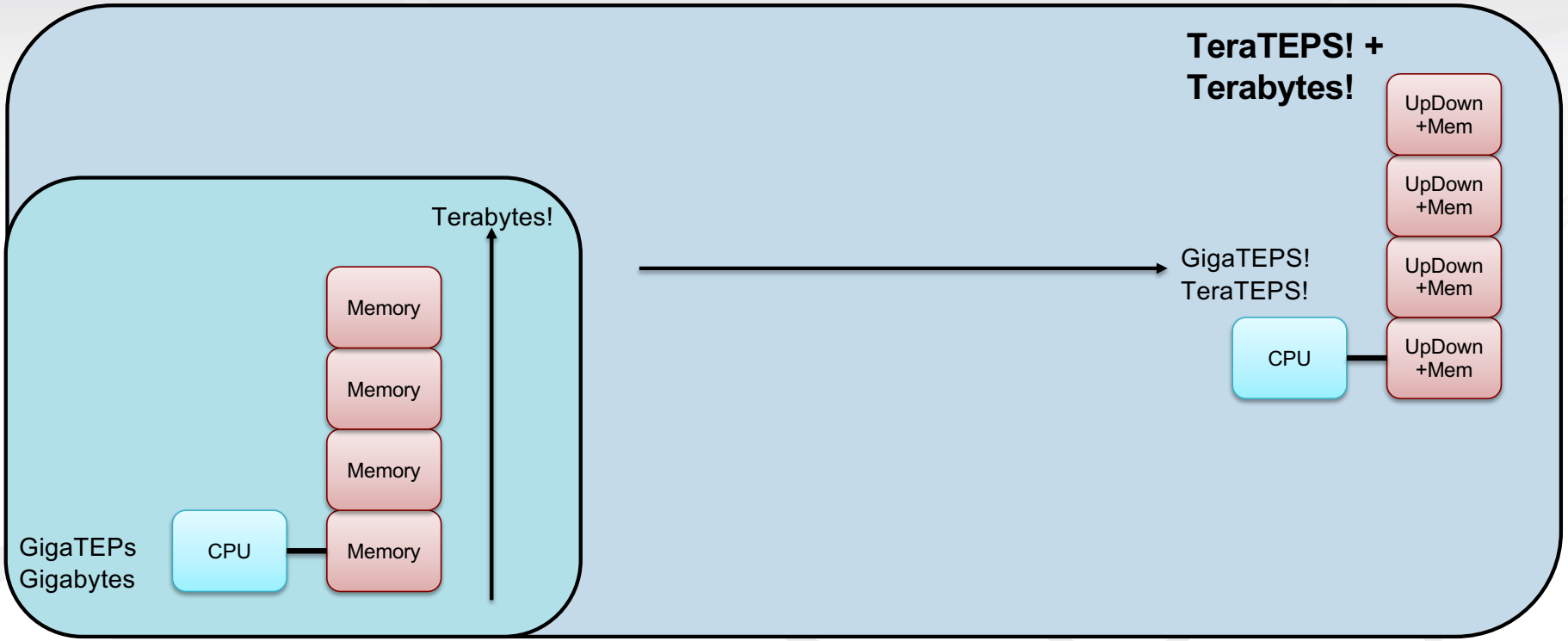



- Built many kernels, workflows
- Compiler, Runtime, Rich Libraries

- BFS: Push, Push-pull, Load-balanced
- PageRank, data-driven PageRank
- K-truss, Triangle Count
- K-core, Louvain
- SCC, CC, 4-cycle
- Influence maximization
- Ingestion and Streaming Queries
- Total Match, Triggered Graph Rematching
- Bioinformatics Contig Assembly
- …

Chicago UpDown

# Graphs need: Memory and Compute




- UpDown enables: compute scaling and memory scaling

Chicago UpDown

# Summary

- UpDown = breakthrough graph computing performance
- Easy to write scalable programs
- Enables scaling compute and memory
  - Single node
  - Large scale system

- Work with us! [www.chupdown.com](http://www.chupdown.com)

Chicago UpDown

# Work with UpDown!

- Graph Applications or Database
  - UpDown's platform provide scaling of your SW performance and value
  - Runs all of your existing SW, exploits acceleration for compute-intensive, bandwidth intensive algorithms
- User, Demanding Graph Applications
  - UpDown creates new levels of performance and capability
  - Seamless path to scale and increase value of your applications

- Hardware platforms available soon!
- Andrew A Chien, aachien@chupdown.com
- http://www.chupdown.com/

Chicago UpDown

# Backup