



**Pacific Northwest**  
NATIONAL LABORATORY

*Proudly Operated by **Battelle** Since 1965*

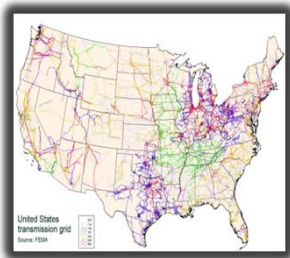
# Graph Analytics in Big Data

John Feo

Pacific Northwest National Laboratory

# A changing World

- ▶ The breadth of problems requiring graph analytics is growing rapidly



Large Network Systems



Social Networks



Packet Inspection



Natural Language Understanding



Semantic Search and Knowledge Discovery



CyberSecurity



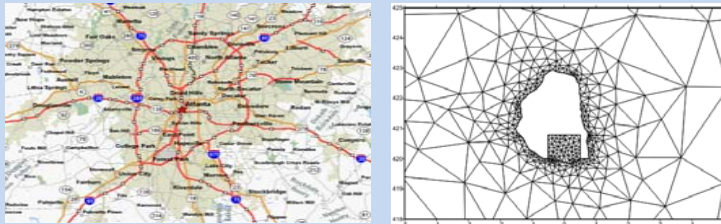
**Pacific Northwest**  
NATIONAL LABORATORY

*Proudly Operated by Battelle Since 1965*

# Graphs are not grids

- ▶ Graphs arising in informatics are very different from the grids used in scientific computing

## Scientific Grids



Static or slowly involving

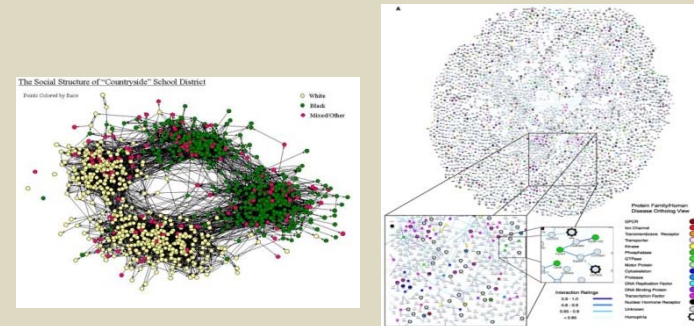
Planar

Nearest neighbor communication

Work performed per cell or node

Work modifies local data

## Graphs for Data Informatics



Dynamic

Non-planar

Communications are non-local and dynamic

Work performed by crawlers or autonomous agents

Work modifies data in many places

# Challenges

- ▶ Problem size
  - Ton of bytes, not ton of flops
- ▶ Little data locality
  - Have only parallelism to tolerate latencies
- ▶ Low computation to communication ratio
  - Single word access
  - Threads limited by loads and stores
- ▶ Frequent synchronization
  - Node, edge, record
- ▶ Work tends to be dynamic and imbalanced
  - Let any processor execute any thread



**Pacific Northwest**  
NATIONAL LABORATORY

*Proudly Operated by Battelle Since 1965*

# System requirements

- ▶ **Global shared memory**
  - No simple data partitions
  - Local storage for thread private data
- ▶ **Network support for single word accesses**
  - Transfer multiple words when locality exists
- ▶ **Multi-threaded processors**
  - Hide latency with parallelism
  - Single cycle context switching
  - Multiple outstanding loads and stores per thread
- ▶ **Full-and-empty bits**
  - Efficient synchronization
  - Wait in memory
- ▶ **Message driven operations**
  - Dynamic work queues
  - Hardware support for thread migration



Cray XMT

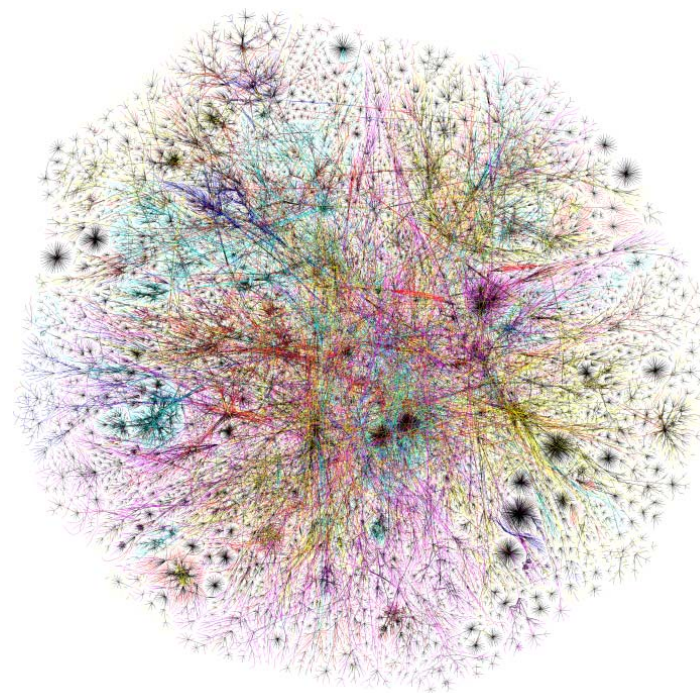


**Pacific Northwest**  
NATIONAL LABORATORY

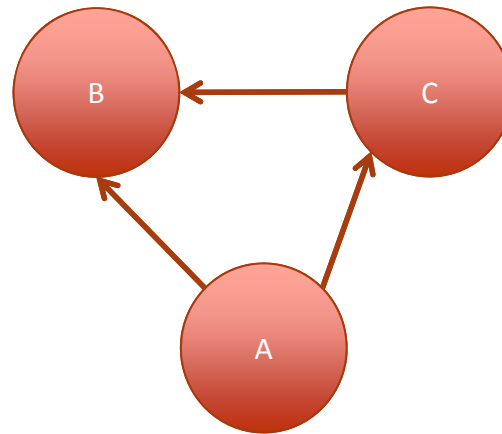
*Proudly Operated by Battelle Since 1965*

# Our type of problems

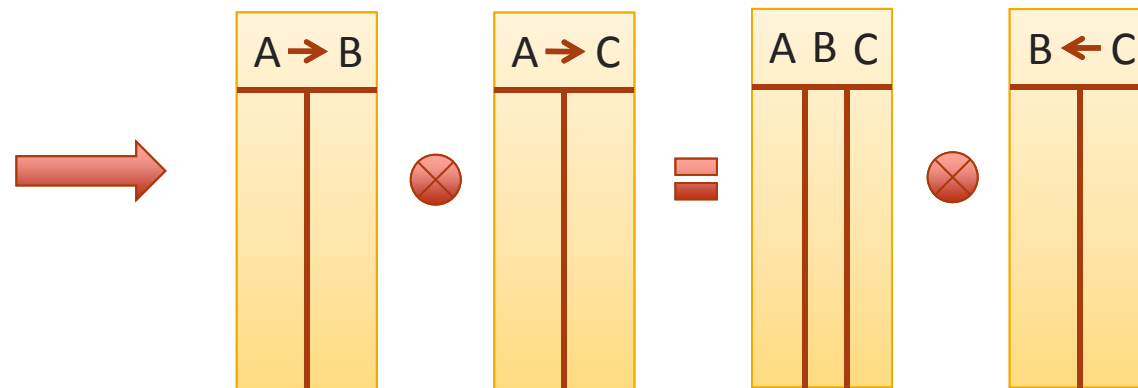
- ▶ Return all triads such that  $(A \rightarrow B), (A \rightarrow C), (C \rightarrow B)$
- ▶ Return all three paths with link types  $\{T_1, T_2, T_3\}$  such that the timestamps of consecutive links overlap by at least 0.5 seconds.
- ▶ From Facebook, return the connected subgraph  $\mathbf{G}(V, E)$  such that  $\mathbf{G}$  includes all the friends of John, the cardinality of  $V$  is minimum, and  $\sum \text{NetWorth}(v_i \in V)$  is maximum.



# Triads



```
SELECT ?A ?B ?C
WHERE { ?A ?a ?B .
        ?A ?b ?C .
        ?C ?c ?B .
}
```



# Simple C code !?!?!

```
for each node A {  
  for each out_edge I of A {  
    for each out_edge J of A {  
      B = tail of I;  
      C = tail of J;  
  
      for each out_edge K of C  
        if tail of K == B {... write answer ...}  
    } } }  
}
```

No memory explosion

15 secs

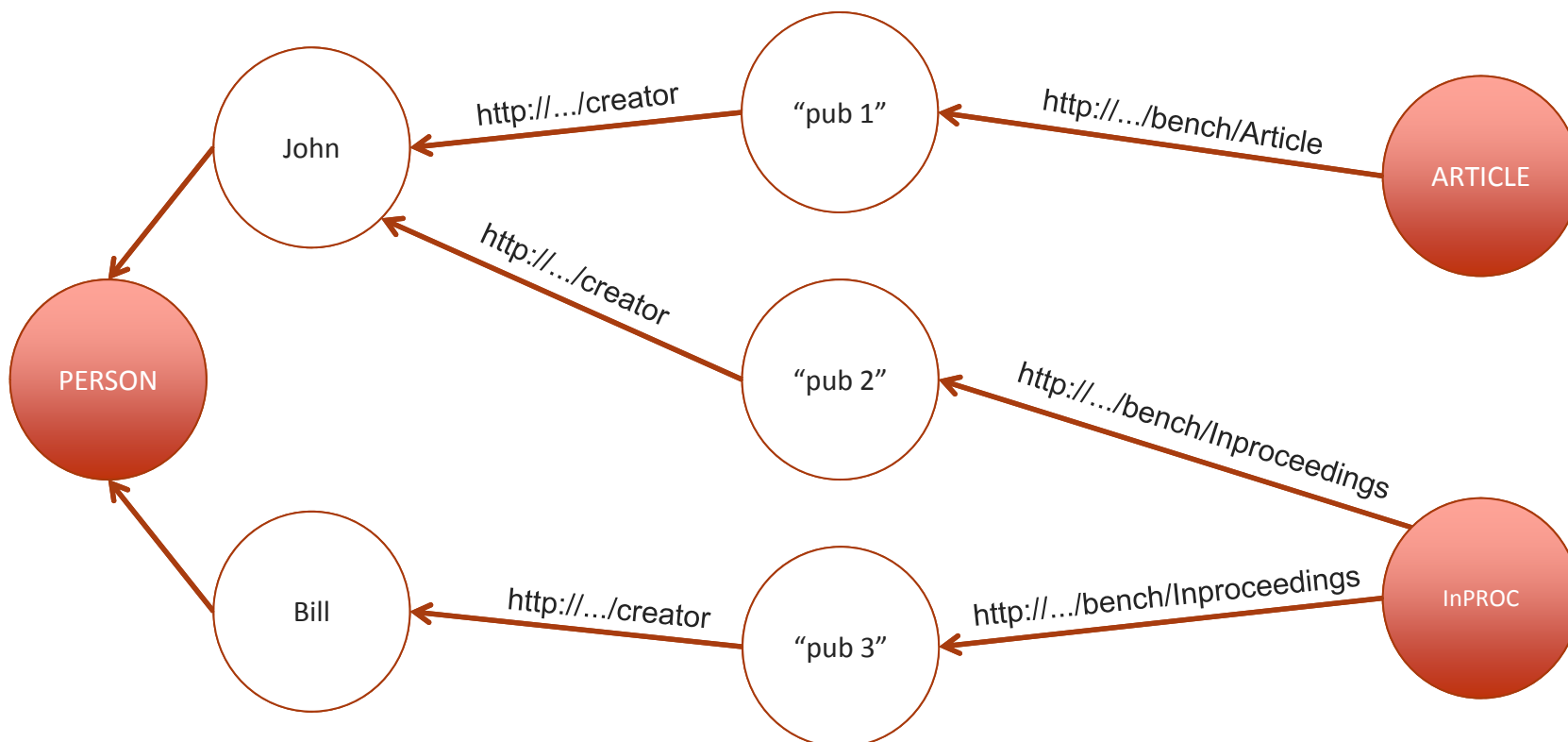


# SP2 Benchmarks

- ▶ We have written the 12 SP2B queries in C using our graph API
- ▶ Execution time on Cray XMT/2 is from **one to three orders magnitude faster** than Virtuoso on 3GHz Xeon server
- ▶ Now porting sdb0 to x86 server and cluster systems
- ▶ C code is simple, but
  - Can we generate it automatically from a high level query language?
  - Can we provide some other more appropriate query interface?

# Query 5

- ▶ Return the names of all persons that occur as author of at least one inproceeding and at least one article



# Data parallel code for Query 5

```
int PERSON_index = get_Vertex_Index(person);
int ARTICLE_index = get_Vertex_Index(article);
int INPROC_index = get_Vertex_index(inproc);

int nibr_Edges = inDegree(PERSON_index);
in_edge_iterator Person_edges = get_InEdges(PERSON_index);

for (i = 0; i < nibr_Edges; i++) {
    int person = PERSON_edges[i].head;
    int nibr_Publ = number_edges(person, creator);
    in_edge_iterator Publ_edges = get_InEdges(person, creator);

    for (j = 0; j < nibr_Publ; j++) {
        int publ_type = edge_Head_Index(Publ_edges[j]);

        if (publ_type == ARTICLE_index) flag |= 1;
        else if (publ_type == INPROC_index) flag |= 2;

        if (flag == 3) {print person; break;}
    }
}
```

**1.29 secs vs. 21 secs in Virtuoso**

# Conclusions

- ▶ Big data graph analytics is fundamentally different than big data science
  - Different algorithms
  - Different challenges
  - Different hardware requirements
- ▶ Conventional database systems based tables and join operations are insufficient
  - Data parallel graph crawls can be orders of magnitude faster
- ▶ Need new query languages capable of expressing graph analytics operations and compiling to data parallel operations