uRiKA
A high-performance multithreaded in-memory graph database appliance

Jim Maltby

uRiKA == universal RDF integration Knowledge Appliance
Outline

- Introduction to uRiKA
- Multithreading
- uRiKA Appliance
- SPARQL Database
- Use Cases
**YarcData uRiKA: Big Data Appliance for Graph Analytics**

Gain business insight by discovering unknown relationships in big data

- **Graph analytics warehouse supports ad hoc queries, pattern-based searches, inferencing and deduction on dynamic data sets**

Hardware and software dedicated to scalable graph analytics

- **Purpose built to solve big data graph problems with large shared-memory and massive multi-threading**

Ease adoption and training with industry standards support

- **Data center ready appliance with open-source interfaces enables re-use of in-house skill sets, standard languages and simplified integration**
Many Big Data problems require Graph Analytics

- Social Networking
- Intelligence/Security
- Telecom/Mobile
- Life Sciences/Biology
- Healthcare/Medicine
- Finance
- Internet/WWW
- Supply Chain
Current architectural approaches may result in low performance on Graph Analytics

Graphs are hard to Partition

High cost to follow relationships that span Cluster Nodes

Network is 100 times SLOWER than Memory*

Graphs are not Predictable

High cost to follow multiple competing paths which cannot be pre-fetched/cached

Memory is 100 times SLOWER than Processor*

Graphs are highly Dynamic

High cost to load multiple, constantly changing datasets into in-memory graph models

Storage I/O is 1000 times SLOWER than Memory I/O*

uRiKA: Purpose-built Graph Appliance…

- Graphs are hard to Partition
  - Large Shared Memory
    - Up to 512 TB

- Graphs are not Predictable
  - Massively Multi-threaded
    - 128 threads/processor

- Graphs are highly Dynamic
  - Highly Scalable I/O
    - Up to 350 TB/hr

Real-time, Interactive Analytics on Big Data Graph Problems
Cray Hardware Engine
- Originally designed for deep analysis of large datasets
- Very large scalable shared memory
  - Architecture can support 512TB shared memory
  - Typical systems are 2 TB to 32 TB
- Multithreading
  - Unique highly multithreaded architecture
  - 128 hardware threads per processor
  - Extreme parallelism, hides memory latency

Multithreaded Graph Database
- Highly parallel in-memory RDF quad store
- High performance inference engine
- High performance parallel I/O

Industry Standard Front End
- Based on Jena open source semantic DB
- All standard Linux infrastructure and languages
- Lustre parallel file system
uRiKA complements existing Data Warehouse/Hadoop environment by offloading Graph Analytics.
Multithreading
Over a decade of innovation and investment

- **MTA-1 – 1998**
  - Gallium arsenide: Proof of concept
  - *First production implementation of latency-tolerant multithreading*

- **MTA-2 – 2002**
  - Transition the microprocessor to CMOS
  - *Invention of scaling memory in a multithreading system*

- **XMT-1 – 2008**
  - Integrated MTA and main Cray product line XT
  - *First hybrid system to support X86 and multithreading CPUs*

- **XMT-2 – 2011**
  - Made system ready for enterprise production
  - *Largest shared memory machine in the world*

- **YarcData uRIKA – 2012**
  - Optimized for Scalable Graph Analytics
Why Multithreading?

- Relative latency to memory continues to increase
  - Vector processors *amortize* memory latency
  - Cache-based microprocessors *reduce* memory latency
  - Multithreaded processors *tolerate* memory latency

- Multithreading is most effective when:
  - Parallelism is abundant
  - Data locality is scarce

- Large graph problems perform well on this architecture
  - Semantic databases
  - Social network analysis
Multithreading Made Simple

- Many threads per processor core; small thread state
- Thread-level context switch at every instruction cycle
Keeping the Bottlenecks Saturated

- **Conventional processor**
  - When one or a few threads stall, memory/network bandwidth become idle.

- **Multithreaded processor**
  - Although some threads stall, others keep issuing local/remote memory requests, keeping most precious resources busy.
Hiding memory latencies

- **Caches**
  - Reduce latency by storing some data in fast, nearby memory

- **Vectors**
  - Amortize latency by fetching $N$ words at a time

- **Parallelism**
  - *Tolerate* latency by switching tasks
  - Multithreading tries to balance “Little’s Law:”
    \[
    \text{concurrency} = \text{bandwidth} \times \text{latency}
    \]
Randomized and Synchronized Memory

Memory addresses are hashed, on cache line (64 byte) boundaries

Full/empty bits on all data words

**sync**
- Data type and HLL functions (readff, readfe, writeef)
- Loads/stores succeed only when full bit is set/unset
- Loads/stores leave full bit unset/set

**future**
- Loads/stores succeed only when full bit is set
- Lazy evaluation and barriers

![Diagram showing tag bits and data values with full-empty traps and forward]

- full-empty
- trap 2
- trap 1
- forward

- 63
- 0

- tag bits
- data values
To the programmer, a multiple processor XMT looks like a single processor, except that the number of threads is increased.
XMT compute board

- 4 DIMM Slots
- Redundant VRMs
- L0 RAS Computer
- CRAY Seastar2™
- CRAY Seastar2™
- CRAY Seastar2™
- CRAY Seastar2™
Cray uRiKA High Density Node

- 4 Threadstorm processors
- 4 memory controllers
- 4 SeaStar2 NICs
- Up to 64 GB per node

Coherent HyperTransport (2 links between sockets)

8 DIMMs Registered DDR2 > 17 GB/sec per node

Threadstorm 4 processors

Cray SeaStar2 Interconnect
uRiKA System Logical View

**Compute**
- MTK

**Service & IO**
- Linux

### Service Partition
- **Linux OS**
- **Specialized Linux nodes**
  - Login PEs
  - IO Server PEs
  - Network Server PEs
  - FS Metadata Server PEs
  - System Server PEs

### Compute Partition
- MTK (BSD)

**Network Components**
- 10 GigE
- Fibre Channel
- RAID Controllers

**PCI-E Connections**
- Network

**10 GigE Connections**
- RAID Controllers
uRiKA is Built on the XT5 Infrastructure

- Uses the same cabinets, boards, scalable interconnect, I/O and storage infrastructure, user environment, and administrative tools...

...just changes the processor

- **Cabinet**
  - 24 blades per cabinet
  - Vertical airflow with optional liquid assist

- **Compute blades**
  - 4 Threadstorm processors
  - 16-64 GB per processor

- **Cray XT service and I/O subsystem**
  - PCIe connections to storage and networks
  - Scalable Lustre global file system

- **Cray XT high-speed 3D torus network**
- Cray XT power and RAS systems
- Linux based user environment
uRiKA Appliance
uRiKA Software / Hardware Stack

- **Linux**
  Apache Tomcat/Jena

- **Graph Analytic Tools**
  Management, Security, Data Pipeline

- **Graph Analytic Database**
  RDF datastore and SPARQL engine

- **Graph Analytic Platform**
  Shared-memory, Multi-threaded, Scalable I/O graph-optimized hardware

- **Data Center** sees another Linux server
- **Applications** see industry standard interface
  - RDF, SPARQL, Java, Gadgets...
- **Reuse Existing Skillsets**
  - Java, OSGI, App Server, SOA, ESB, Web toolkit...
- **Standard languages**
  - All applications and artifacts built on uRiKA can be run on other platforms
SELECT ?p ?x
WHERE {
  (?x type person)
  (?p sells "DVD")
  (?x shops-at ?p)
}
SPARQL Database
RDF triples databases are inherently graphical

Some researchers call semantic databases “semantic graph databases”
Emerging Web 3.0 Standards: RDF and SPARQL

Resource Description Framework (RDF)
- Designed to enable semantic web searching and integration of disparate data sources
- W3C standard formats
- Every datum represented as subject/predicate/object
  - Ideally with each of those expressed with a URI
- Standard ontologies in some domains
  - *e.g.*, Open Biological and Biomedical Ontologies (OBO)
- Examples:

```xml
<nctbitax:NCBITaxon_840261> rdf:type owl:Class
<nctbitax:NCBITaxon_195644> rdfs:subClassOf <ncbitax:NCBITaxon_185881>
<nctbitax:NCBITaxon_816681> rdfs:label "Characiformes sp. BOLD:AAG5151"@en
```
EMERGING WEB 3.0 STANDARDS: RDF AND SPARQL

- **SPARQL Protocol and RDF Query Language (SPARQL)**
  - Enables matching of graph patterns in the semantic DB
  - Reminiscent of SQL

---

# Lehigh University BenchMark (LUBM) Query 9

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>

WHERE
{?X rdf:type ub:Student .
 ?Y rdf:type ub:Faculty .
 ?Z rdf:type ub:Course .
 ?X ub:takesCourse ?Z}
Use Cases
uRiKA Customer Use Case: Cancer Research Institute

Discover new drugs and treatments

The Challenge
- Multiple massive datasets describing biological network graphs in cancer cells from published literature and experimental data, constantly updated
- Un-partitionable, densely and irregularly connected graphs
- Multiple researchers concurrently searching for relationships not found in published literature

uRiKA Solution
- uRiKA holds un-partitioned genomic network graph in memory
- Contrast experimental models and theories with published results to discover previously unknown relationships
- Interactive, real time access by multiple researchers

Business Value
- Identify new pathways in cell models to refine cancer treatments

Confirmation of elevated VEGF levels by tissue microarray:
Biotech researchers have mapped most of the basic building blocks of genes – but the interactions are not understood
• "Parts List for the human body, but where’s the Haynes manual"

Biotech researchers are constantly looking for new and non-obvious interactions between genes and protein families.

Medical research churns out massive quantities of literature, but most researchers don’t have time to read even their own specialty publications
• Over 10,000 genomics publications per month currently
The Technical challenge

- The Cancer Genome Atlas (TCGA) data combined with literature relationships from Medline
  - Protein domain interactions are extracted from TCGA using Random Forest classification
  - A Normalized Medline Distance (NMD) between proteins is calculated from Medline titles and abstracts
  - Protein data from Uniprot and Pfam are also included

- 1.2 billion entries that generate 6B triples
  - New data sets constantly being added - mutation data, patient data, methylation data, statistical analysis results and more...

- Current database technology does not have the performance for useful full-scale queries across this data
Discover “similar” Patients to optimize treatment

The Challenge

- Longitudinal, historical data spanning all events, symptoms, diagnoses, diseases, treatments, prescriptions, etc of 10M patients including genetics and family history
- Ad-hoc, constantly changing definition of “similarity” based on thousands of parameters
- Interactive, real-time response during consultation

uRiKA Solution

- uRiKA holds entire relationship graph in memory
- Identify “similar” patients based on ad-hoc physician specified patterns
- Interactive, real-time access by entire physician community

Business Value

- Consistent selection of the most effective treatment for each patient by each doctor every time
Thank you!

James D. Maltby, Ph.D.
jmaltby@yarcdata.com