rough title:

KM for use in SmartThings projects

was:

Large Linked Data

Sketch of a potential *linked-data* system that could deal w/large data volumes

w/example influences/components

bit.ly/TP8gfz

Large Triple/Graph Processing

dealing with communicating distributed stores and streaming data feeds

more of a Ipython/parallel /ROS approach

making a *BBN-SHARD* like system using *UT-Austin's Knowledge-Machine* in Lisp

mike.bobak.googlepages.com
bobak7.wix.com/bobak

Possible app: mapreduce based graph-store

by bringing together ideas behind BBN's SHARD mapreduce on hdfs flat files of triples, &/or using aspect of Ipython's parallel-libs, Storm, &ROS with the reasoning done by UT-Austin's Knowledge-Machine and SciDB like array storage/ops for heavy numerics

SHARD

Raytheon BBN Technologies

Graph processing on HDFS

fast distributed triple-store

SciDB

SciDB

open-src data-mgt&analytics

array-ops scaling to petabytes

KM: The Knowledge-Machine

frames to OWL, use of access-limited-logic

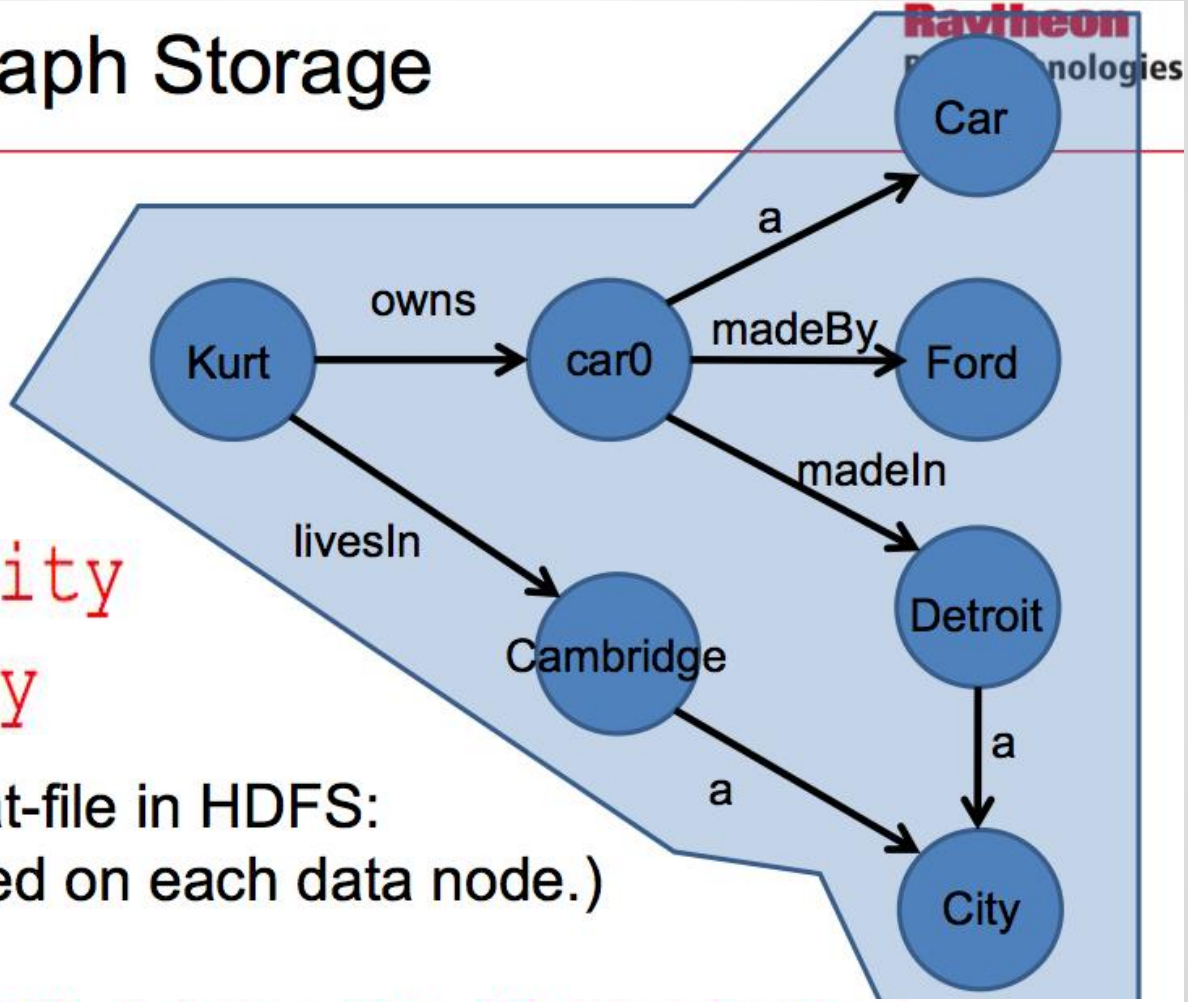
KM is a powerful, frame-based language with clear first-order logic semantics. It contains sophisticated machinery for reasoning, including selection by description, unification, classification, and reasoning about actions using a situations mechanism.

Used in Halo/Calo precursor to Siri

I used for medical free-text concept-labeling experiments, along w/MMTx/MetaMap POS/..

SHARD: (Scalable, High-Performance, Robust and Distributed)

HDFS Graph Storage




```
Cambridge a City
Detroit a City
```

Graphs saved as flat-file in HDFS:
(Portions of file saved on each data node.)

```
Kurt owns car0 livesIn Cambridge
Car0 a Car madeBy Ford madeIn Detroit
```

graph/triple-store processing via mapreduce

BBN's SHARD triple-store uses flat HDFS files w/great benchmarks. I think this could be done w/Lisp, and have considered doing some of the reasoning w/KnowledgeMachine, which was used in the HALO, then CALO project which turned into Siri.

SHARD http://www.lotico.com/slides/20110208/Rohloff_Meetup_02_08_2011.pdf

-but w/KnRep&Reasoning done w/:

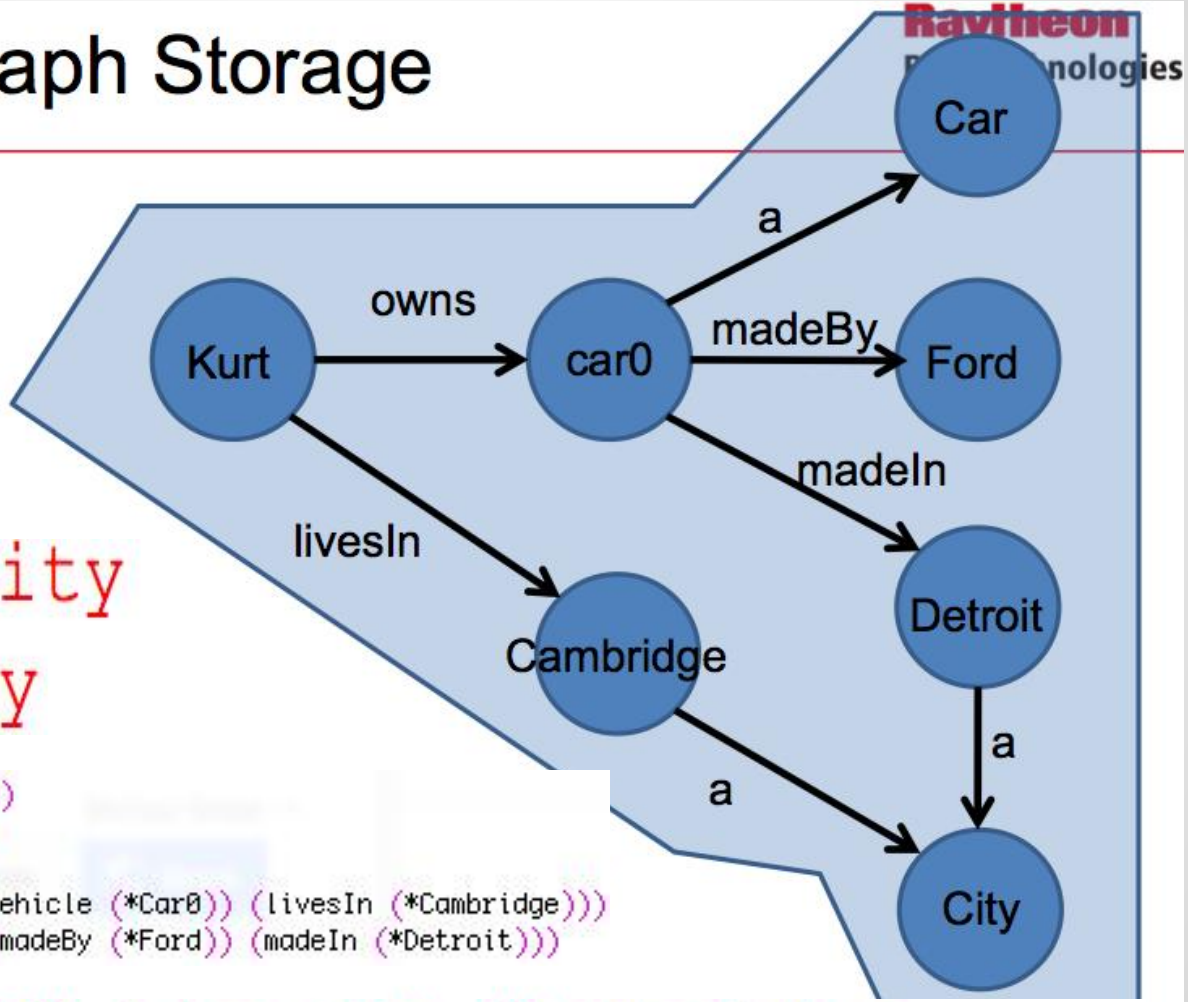
KM http://www.cs.utexas.edu/~mfkb/km/ example class-defn:

```
;;; "Cars are vehicles."
;;; superclasses(Car, Vehicle) ∧ terms(Car,"car") ∧ terms(Car,"vehicle")
(Car has                          ; properties of the class
  (superclasses (Vehicle))
  (terms ("car" "automobile"))) ; ie. words of phrases used to denote cars
```

```
;;; "Cars have four wheels, an engine, and a chassis"
;;; ∀c isa(c, Car) → wheel-count(c, 4)
;;; ∀c isa(c, Car) → ∃e, ch isa(e, Engine) ∧ isa(ch, Chassis) ∧ parts(c, e) ∧ parts(c, ch)
(every Car has
  (wheel-count (4))
  (parts ((a Engine) (a Chassis))))
```

SHARD: Using Hadoop to Build a Scalable, Distributed Triple Store

HDFS Graph Storage



Cambridge a City

Detroit a City

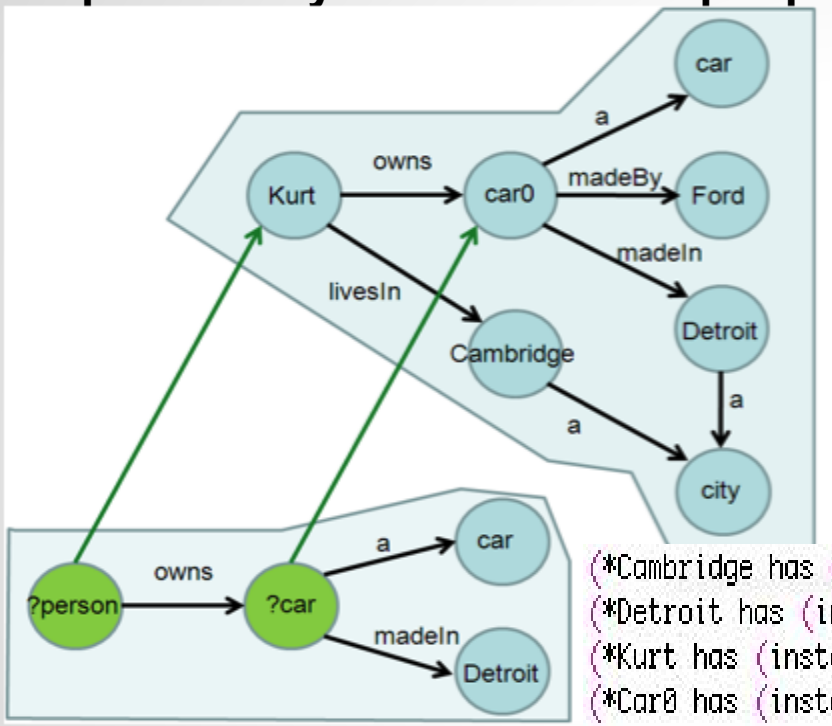
```
(*Cambridge has (instance-of (City)))
(*Detroit has (instance-of (City)))

(*Kurt has (instance-of (Person)) (vehicle (*Car0)) (livesIn (*Cambridge)))
(*Car0 has (instance-of (Vehicle)) (madeBy (*Ford)) (madeIn (*Detroit)))
```

Kurt owns car0 livesIn Cambridge

Car0 a Car madeBy Ford madeIn Detroit

Shard sparql query w/KM equivalent

https://dist-systems.bbn.com/people/krohloff/shard_overview.shtml



SRI has OWL interop

So might also w/sparql

```
(*Cambridge has (instance-of (City)))
(*Detroit has (instance-of (City)))
(*Kurt has (instance-of (Person)) (vehicle (*Car0)) (livesIn (*Cambridge)))
(*Car0 has (instance-of (Vehicle)) (madeBy (*Ford)) (madeIn (*Detroit)))
```

```
SELECT ?person
WHERE {
?person :owns ?car .
?car :a :car .
?car :madeIn :Detroit .}
```

```
(Car has (superclasses (Vehicle)))
;when vehicle predicate inherits from owns can say:
 (the  owns-of of (a Car) with ((madeIn *Detroit)))
;or just ask:
(the  vehicle-of of (a Car) with ((madeIn *Detroit)))
```

Streaming is very flexible

Allowing polyglot programming (can mix in elts in any language needed)

Hadoop Streaming

Hadoop provides an API to MapReduce that allows you to write your map and reduce functions in languages other than Java. *Hadoop Streaming* uses Unix standard streams as the interface between Hadoop and your program, so you can use any language that can read standard input and write to standard output to write your MapReduce program.

Streaming is naturally suited for text processing (although, as of version 0.21.0, it can handle binary streams, too), and when used in text mode, it has a line-oriented view of data. Map input data is passed over standard input to your map function, which processes it line by line and writes lines to standard output. A map output key-value pair is written as a single tab-delimited line. Input to the reduce function is in the same format—a tab-separated key-value pair—passed over standard input. The reduce function reads lines from standard input, which the framework guarantees are sorted by key, and writes its results to standard output.

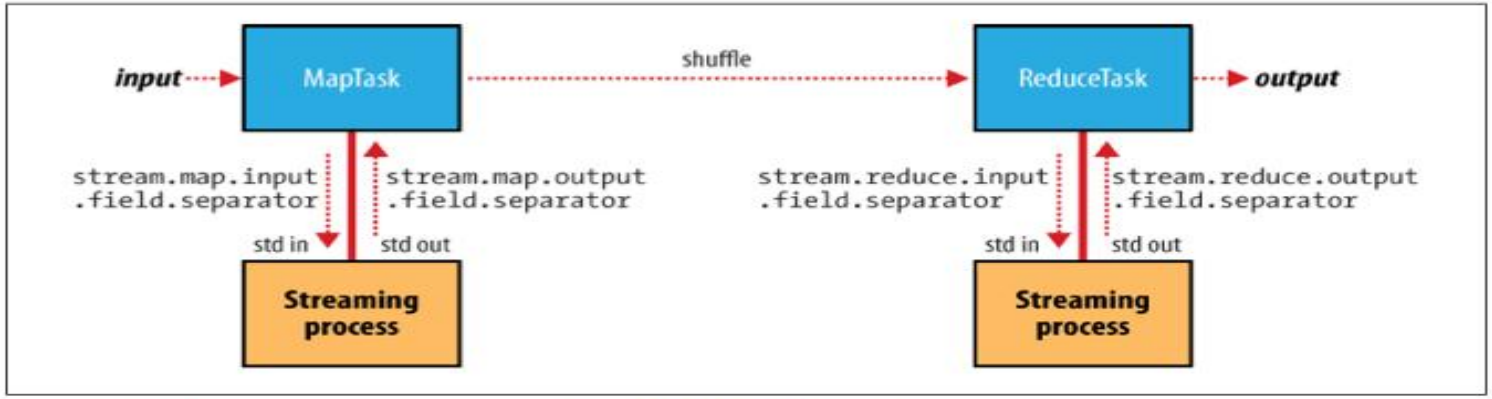


Figure 7-1. Where separators are used in a Streaming MapReduce job

example streaming setup files

```
hadoop fs -rmr streaming/results
hadoop jar /usr/lib/hadoop/contrib/streaming/hadoop-streaming-0.20.2.jar \
    -verbose \
    -input streaming/input-dir \
    -output streaming/results \
        -mapper  "sbcl --core km.core --load mapper.cl" \
        -reducer "sbcl --core km.core --load reducer.cl" \
    -file mapper.cl \
    -file reducer.cl \
    -file km.core
```

Other types of Streaming

Instead of batch Hadoop, there is Storm like live streaming, &/or Ipython's parallel libs, both of which use MQ.

There are also streaming sparql engines like C-SPARQL and Instans, and just streaming triples into your reasoner, which can deal w/updated data.

Keeping it distributed and closer to pub/sub msgs of interest; Like through use of ROS

Clojure (cascalog&storm) (JVM v CL) interop

A *Storm* like system could help deal with some of the *velocity problem*.

Can get KM in ABCL, might mix w/clojure vs a rewrite.

Look great, but I'd rather build up in Lisp, and move it toward what Clojure can do, vs. not building on the rich set of Lisp Libs.

Could probably use cl-zmq as a starting place for a storm-like.

I wish java interop was easier, Rich Hickey tried it before writing Clojure.

I think external-program, streaming,&msg-passing can get around some of the/se tighter/(more radical) ways of getting interop.

Remember your freedom to choose which (set of) language(s) to use should be increased (both in server&) distributed programming, given this looser (stream/msg-passing/..) coupling.

Can also interact w/AWS ..

Can get to S3 storage w/: www.xach.com/lisp/zs3/ get-string/put-string &share w/other apps, via: common-lisp.net/project/cl-json/ strings

or

Simple Command-Line Access to Amazon EC2 and Amazon S3, via:

common-lisp.net/project/trivial-shell/ to aws.amazon.com/developertools/739 AWS shell tool.

NoSQL fast front-end use by puting the json to www.cliki.net/cl-mongo &can make use of: cliki.net/cl-redis too.

Had considered being able to take avro (self-describing) inputs.

It has a json header that describes the binary encoded stream.

can even mix w/hadoop-streaming, but be sure that it is faster than a multi-step process, which could be organized&run by oozie

Though for some tasks you might just be able to use *cascalog*, in clojure which allows for tighter Hadoop/Java integration,

&for more real-time streaming work a clojure hadoop like system called *storm* which uses MQ,

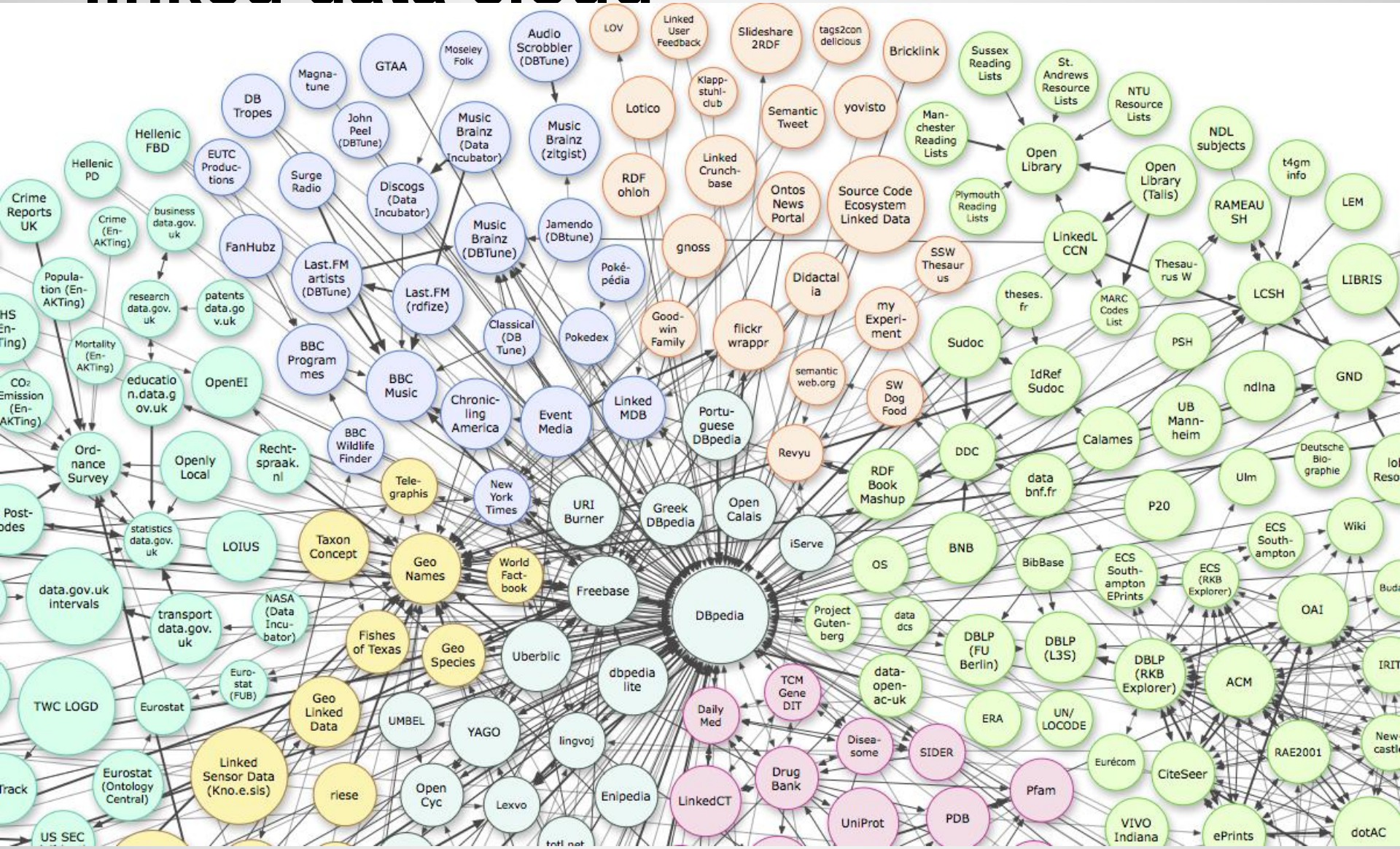
though possibly rewritable in Lisp.

New: will look at: https://github.com/lookis/service-monitor

Example use: bit.ly/hurricanehackers-gdoc tasks using logd.tw.rpi.edu linked-open-govt/etc data

#HurricaneHackers

linked data cloud



Reasons for triples describing large arrays vs. a triple per #:
few extra orders of magnitude by directing data to right tools

Plenty of large numeric needs:
sensor-nets, large-science (weather/satellite-imagery..)
machine-learning, query over huge multi-dim arrays

HPC techniques&tools for store/moving&processing these volumes very quickly.

HDF5 w/parallel (msg-passing) IO &tools that use it.

Links: www.sbcl.org or ccl.clozure.com & www.quicklisp.org

www.cs.utexas.edu/~mfkb/km KnRepr&Reasoning/Qry

www.dist-systems.bbn.com/people/krohloff/shard.shtml

also see the Lisp: (pay): http://franz.com/agraph/ (&free:) https://github.com/kraison/vivace-graph-v2.git

Data: code.google.com/p/cl-protobuf/ or avro.apache.org/ in lisp w/cl-json &a decent cl binary pkg *I'd like semantics for*

Store: www.hdfgroup.org/HDF5/ git://gitorious.org/dh-misc/hdf5.git github.com/filonenko-mikhail/cl-scidb.git <- *petabytes,*

- parallel/dist/msg-passing: lparallel.com cl-pvm/lpvm www.cliki.net/cl-mpi www.csm.ornl.gov/pvm/PVMvsMPI.ps

&miss: http://en.wikipedia.org/wiki/*Lisp 's pvars: auto-spread vects among processors

some Lisp(likes): en.wikipedia.org/wiki/CLIPS (has CLOS) RuleBased-Shell

In CL there is: http://lisa.sf.net/ .. gui: protege.stanford.edu/

Streaming SPARQL: github.com/aaltodsg/instans

ML: code.google.com/p/malecoli/ github.com/mathematical-systems/clml.git http://quickdocs.org/mgl/

en.wikipedia.org/wiki/XLispStat or in CL:github.com/blindglobe/common-lisp-stat

www.visualstats.org lush.sf.net common-lisp.net/project/rcl to R which has

or cliki.net/Mathematics R+pvm: cran.r-project.org/web/packages/rpvm/index.html

common-lisp.net/project/cl-octave &for py pkgs github.com/franzinc/cl-python.git to octave.sf.net

ipython is allowing mixing R/etc, & parallel execution, *sage* mixes python &maxima in Lisp

new: https://github.com/Paradigm4/SciDBR &v7: https://github.com/openlink/virtuoso-opensource *more

github.com/nathanmarz cascalog storm in Clojure

Cascalog is a replacement for tools like Pig,Hive,&Cascading.

Storm is a distributed, reliable,&fault-tolerant stream processing system.

MsgPassing:PVM (a more simple MPI)

We are seeing HPC *msg-passing* libs like MPI being brought into new Hadoop like packages; to allow for *orders of magnitude improvements* (via tigher-loops) for applications like *Machine-Learning*

Having *distributed-objects* w/huge arrays in Lisp is very easy, as you have the ability to easily reflect during runtime; slot-names & associated msg-packing routines for all their slot-value types make it easy. The only trick was to start the obj-packed msg, w/a str, that could be eval'd to create an instance on the other side. Then assoc msg-unpacking methods for each of the slot/val pair comes into play. T can be a nice way to quickly share huge chunks of data between several different dynamic languages.

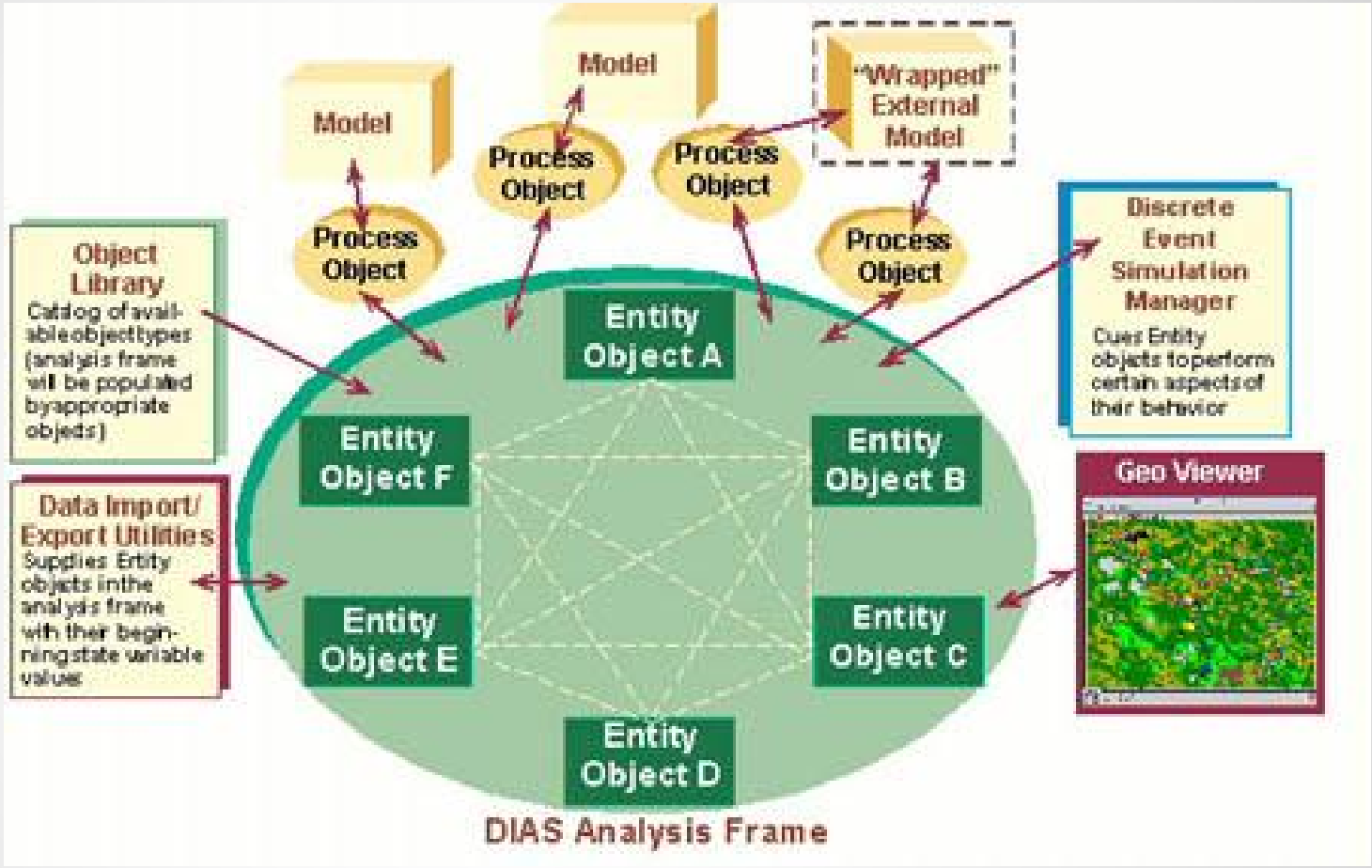
Even tighter coupling like in rcl&cl-octave can benefit from msg-passing;

could maybe re-do w/pvm &(a link to the server version of) R&Octove's REPLs, or just unpack&eval.

I would also like to share lots of data w/xlispstat's vista, Lush,&Clojure

A live:data/goal-driven distrib-simulation

used in:



reasoning/comm

for Context/SimMngr at: https://github.com/MBcode/CLIPSmsc/tree/master/clp-pvm

cl-pvm also available, so I would write similar methods, so they could interoperate.

Semantic-Web/.. to aid Machine-Learning
Adding explicit model/structure, can aid in more causal/explanation/understanding/trust/usability/..

Now I would bring more KnShareingEffort like annotation around large data-sets and code(sim/analytics/etc), so they could more easily interoperate in a goal-based way. So extending old work, but doing it w/(hopefully)a better set of tools.

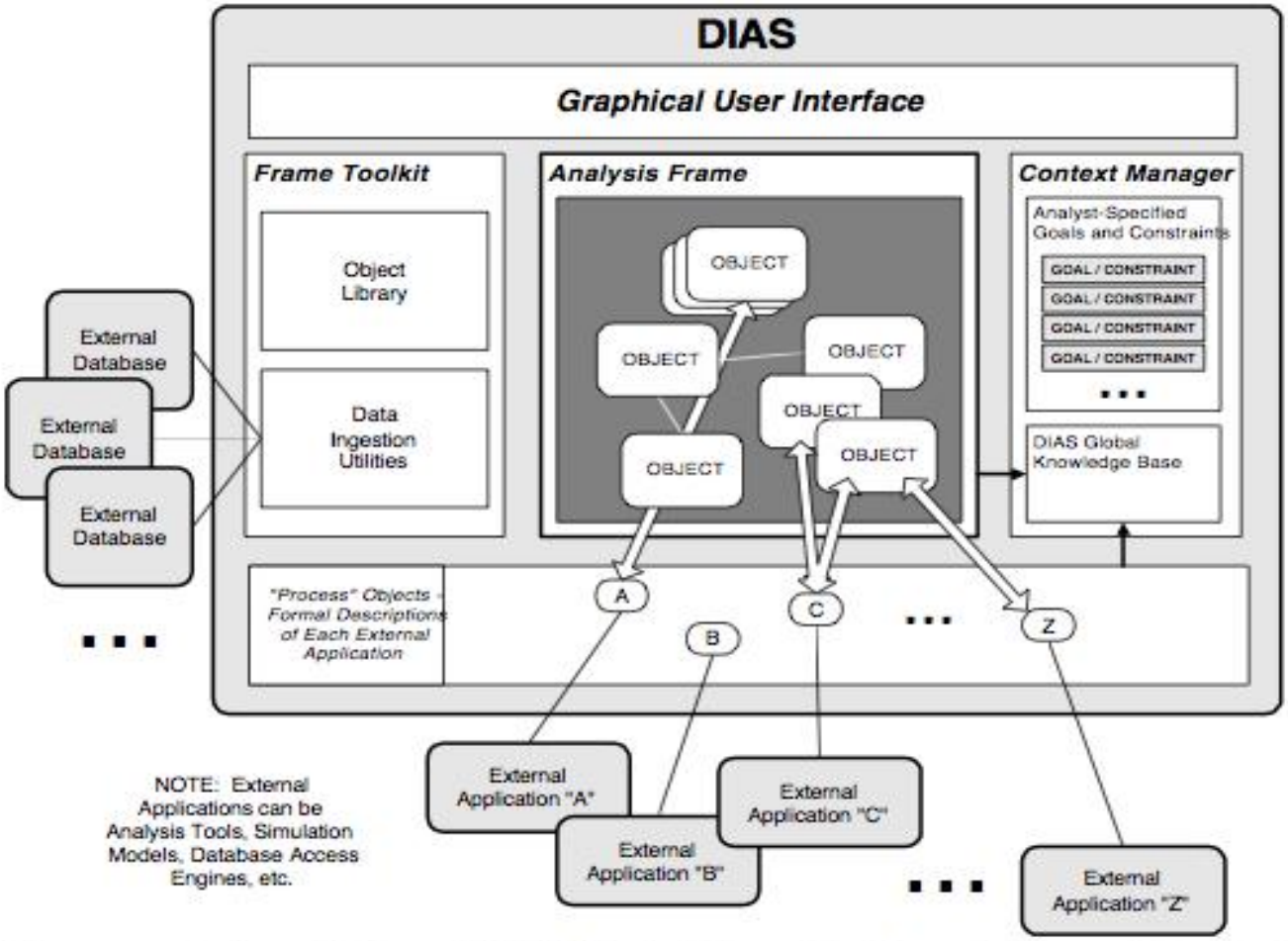


Figure 3. Representation of How DIAS Links User Goals and Constraints to DIAS Resources, Such as Objects, Applications, and Knowledge Bases

*more I'll add a few more new links here:

Many can be found on my: https://twitter.com/MBstream &some others:

http://thinkaurelius.com/2013/05/13/educating-the-planet-with-pearson/

http://www.openlinksw.com/dataspace/doc/oerling/weblog/Orri%20Erling's%20Blog/1728

http://www.franz.com/ps/services/conferences_seminars/semantic_technologies_v27.lhtml

from: http://www.franz.com/ps/services/conferences_seminars/ using: http://www.franz.com/agraph/allegrograph/

http://semanticweb.com/big-data-goes-to-the-ballpark-the-next-generation-of-moneyball-at-yarcdata_b37840

http://portal.utpa.edu/utpa_main/daa_home/coecs_home/provbase_home

Will try bits of this for the http://lispinsummerprojects.org/ at: https://github.com/MBcode/kme

EDU: http://linkedup-project.eu/

NLP: https://github.com/MBcode/km3 http://www.cliki.net/natural%20language%20processing

LinkedData: benchmark: http://swat.cse.lehigh.edu/projects/lubm/

another cloud triplestore: http://sqrrl.com/media/Rya_CloudI20121.pdf