



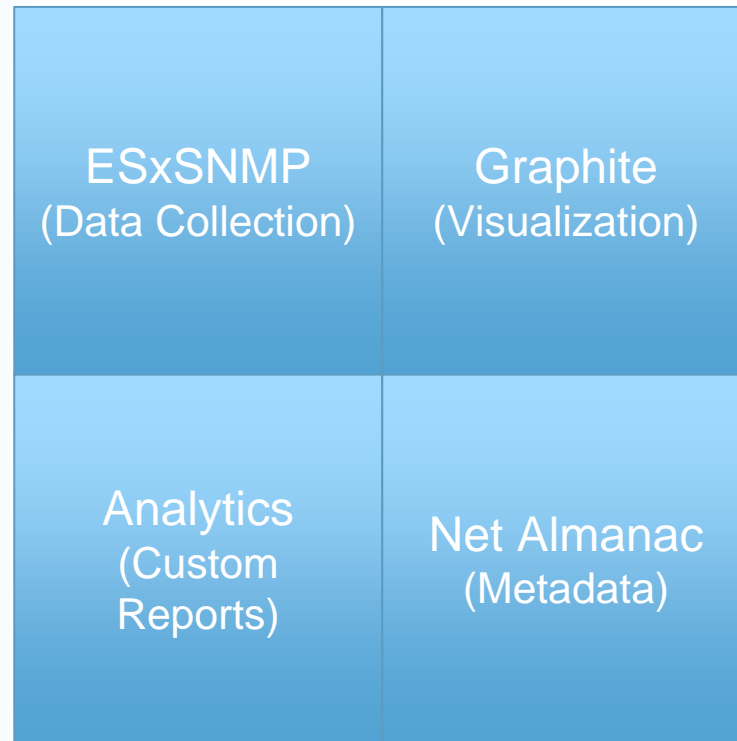
ESxSNMP: ESnet eXtensible SNMP System

Jon Dugan <jdugan@es.net>

Summer JointTechs 2010, Columbus, OH



ESnet Statistics Overview





Why another SNMP collection system?

Desire to retain raw, unmolested data

- Disk is cheap
- At least one application in ESnet needs raw data
- Aids in troubleshooting of erroneous statistics

Automatically detect changes in the network

- New interfaces are detected within minutes (tunable)
- Dynamic circuits make this especially important

Metadata

- Efficiently maintain a history of the network

Integration with other systems

- perfSONAR, web sites (REST/JSON)

Principles



Automate

- routers are added/removed automatically
- interfaces are added/removed automatically
- monthly reports are generated automatically

Don't make (too many) assumptions

- keep raw data
- keep history
- polling does minimal processing/interpretation

Be flexible

- work in the general case, allow for exception cases
- group of cooperating processes, components can be replaced
- clear entry points for modifying behavior

Share

- REST/JSON
- perfSONAR (coming soon)

Implementation



Python

- About 4,000 SLOC
- Dependencies
 - TSDB
 - SQLAlchemy
 - web.py
 - DLNetSNMP: wrapper for net-snmp



Multiple Process Architecture

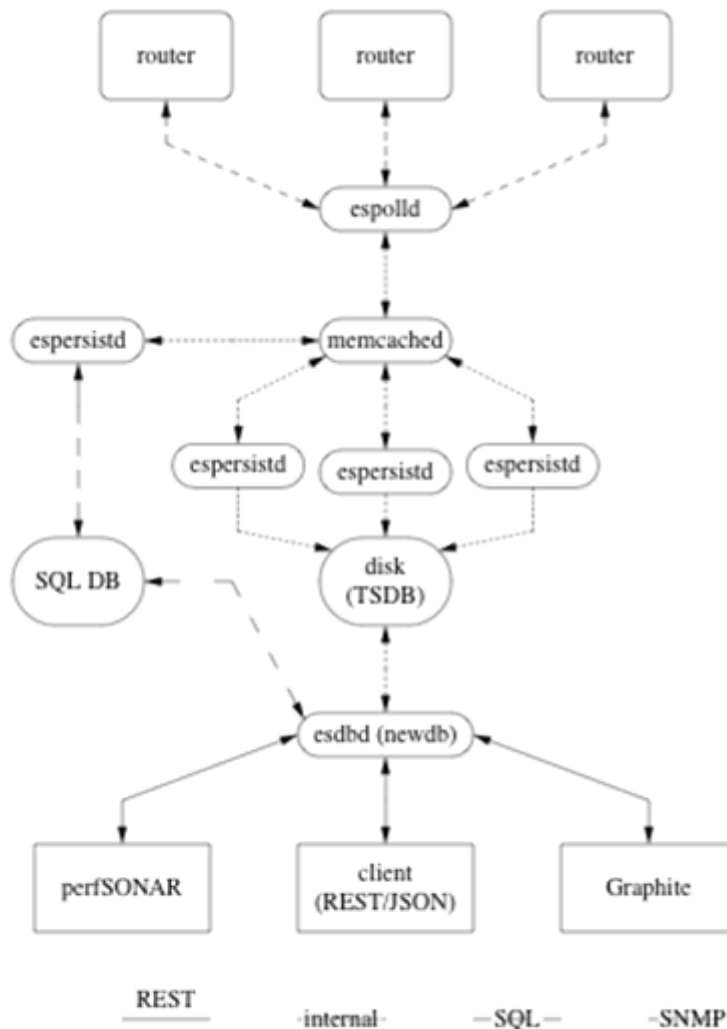
Flexibility / Reduced Coupling

- Components can be replaced individually
- One part can be upgraded without restarting everything
- Each piece is simple and focused (cf. Unix tools approach)

Concurrency

- Process level concurrency
- “Do not communicate by sharing memory; instead share memory by communicating.” (Go language)
- Python threads are lacking
 - Excellent for I/O bound applications
 - Poor performance for compute intensive applications

ESxSNMP Architecture





Devices, OIDs and OIDSets (Oh my!)

Device

- Referred to by DNS name
- Properties: SNMP community, begin time, end time

OID

- Referred to by shortest unique name, eg ifHCInOctets

OIDSet

- Collection of OIDs that are polled as a group
 - FastPollHC: ifHCInOctets, ifHCOctets
 - ifRefPoll: ifDescr, ifAlias, ifSpeed, ifHighSpeed, ifIndex, ipAdEntIfIndex



Polling: espollid

Multithreaded process

- thread per OIDSet per Device
- Thread to gather results and ship them to work queue
- (It's I/O bound so Python threads are happy)

Polling method

- Call correlator setup function (might do get some additional info)
- For oid in oidset: get entire table (BULKWALK)
- Correlate results
- Hand off to thread which will place them in the work queue



Correlators

Provide a mapping between raw data and something useful

Examples:

- IfDescrCorrelator
 - translates ifIndex to ifDescr (interface name)
 - ifHCInOctets.116 becomes ifHCInOctets/ge-1_0_0
- JnxFirewallCorrelator
 - jnxFWCounterByteCount."test-from-eqx"."from-eqx".counter becomes counter/test-from-eqx/from-eqx



Work Queue

Advantages

- Decouple polling from saving to disk (persisting)
- Distribute the load
- Send to multiple work queues (if there are multiple consumers)
- Use all cores without thrashing

Implementation

- Currently memcached and custom code
- Work objects represented as JSON strings
 - Anything that can do JSON and talk to memcached can enqueue work (Perl, Ruby, C/C++, etc, etc)
- AMPQ is a possible future alternative



Metrics Storage: `esperistd`

Manager process

- Starts/stops/restarts worker processes

Worker processes

- Simple ones have single instance
- Compute and I/O bound tasks have many worker processes
 - In this case n workers are started
 - Queuing process hashes them into a worker specific queue
 - (Service, `OIDSet`) tuple hashed to same worker queue each time

Lookup table maps `OIDSet` to a class that handles the persisting



TSDB: Time Series Database

Custom data store, similar to RRD without the round robin

- Never discards raw data
- Generates aggregates

Basic Types

- TSDB
 - Container for TSDBSets and TSDBVars
 - Implements a simple multilevel file store
- TSDBSet
 - It's a directory/folder
 - Container to hold TSDBVars or TSDBSets
- TSDBVar
 - Contains actual data
 - Stored in chunks (usually one day of data)
 - Each var is made up of rows which are indexed by timestamp

TSDB Aggregates



Average / Minimum / Maximum per time interval

Stackable time periods

- Aggregates can be used to generate larger time period aggregates
 - eg. 30 second, 5 minute, 1 hour, 24 hour, 1 week

Aggregate calculation is very similar to RRD

- Test suite includes comparison tests with RRD
- Within 2% to pass (floating point can be a harsh mistress)



Metrics Retrieval: esdbd

Allow easy, consistent access to data

- Data will be used in unanticipated ways
- Language neutral

Technical details

- RESTful interface
- URL hierarchy: eg, `core-rtr-1/interface/xe-0_0_0/in`
- HTTP transport using HTTP semantics
- Data returned in JSON format
- Uses web.py framework

perfSONAR Support



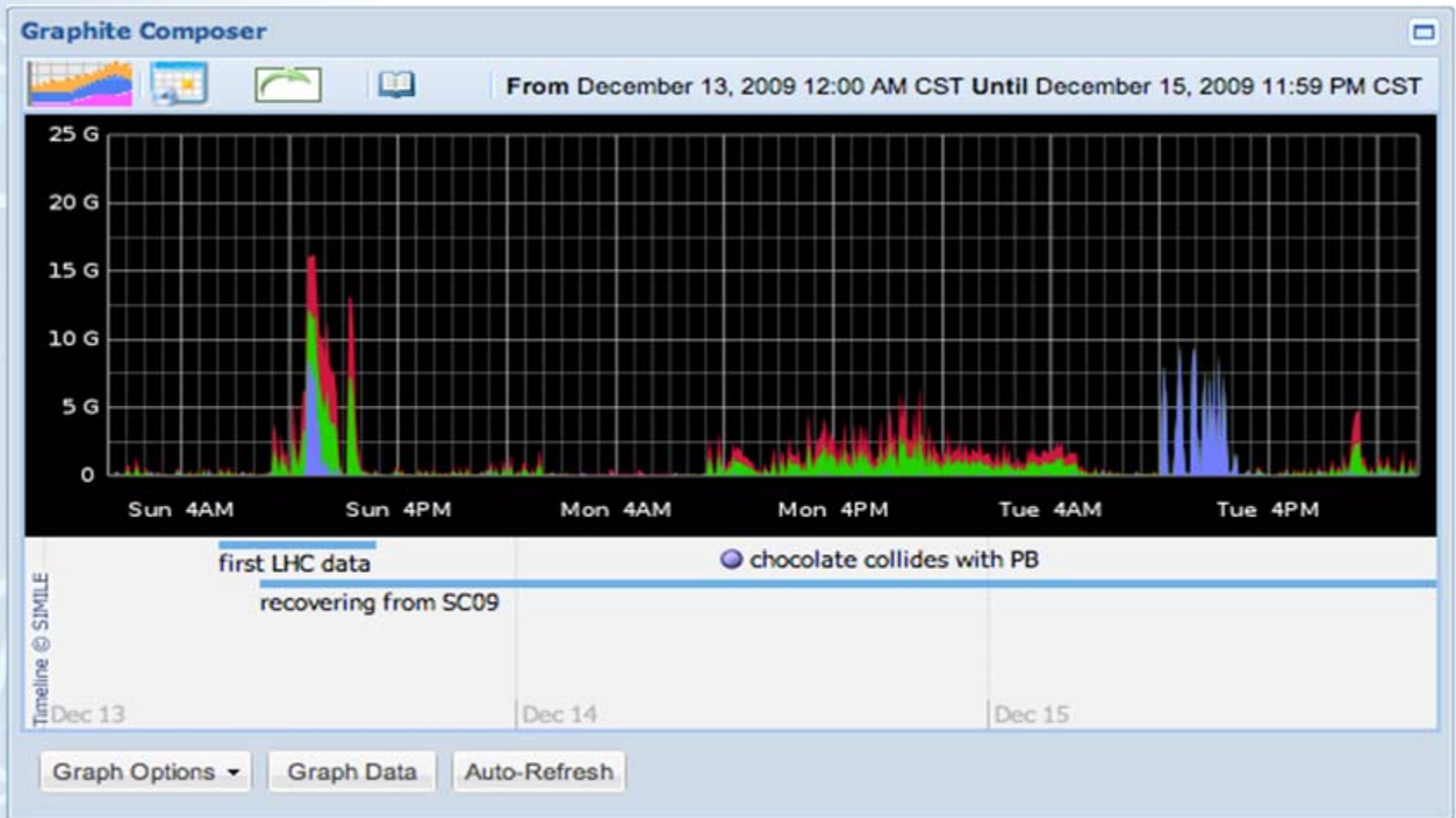
Current

- Added a new database type to the perfSONAR-PS SNMP MA
- Uses the esdbd REST interface
- In production for at least a year

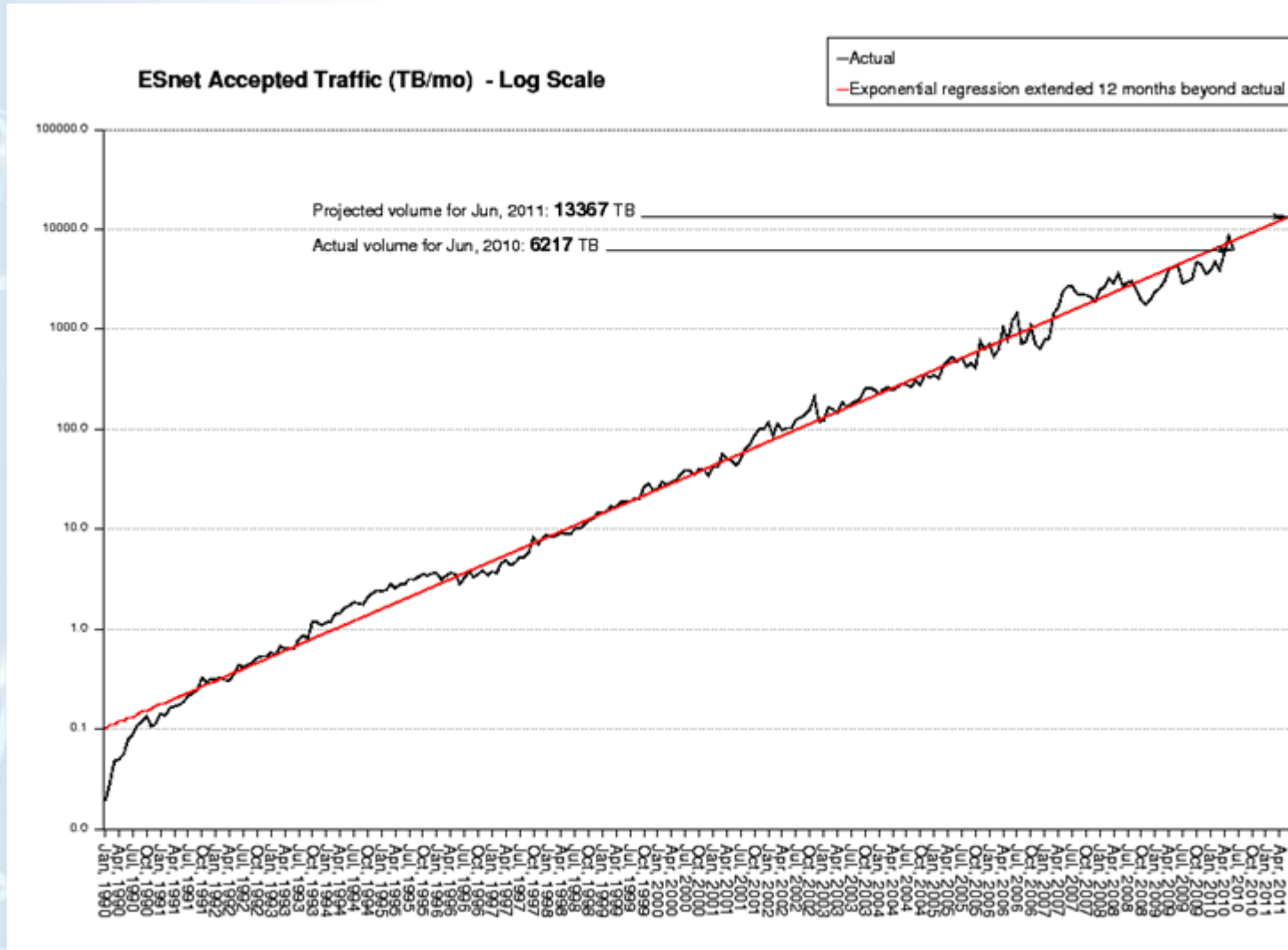
Future

- esdb will be a perfSONAR-MA
- Some initial exploration done already
- Initial Python perfSONAR library developed by Dan Gunter and Monte Goode at LBL

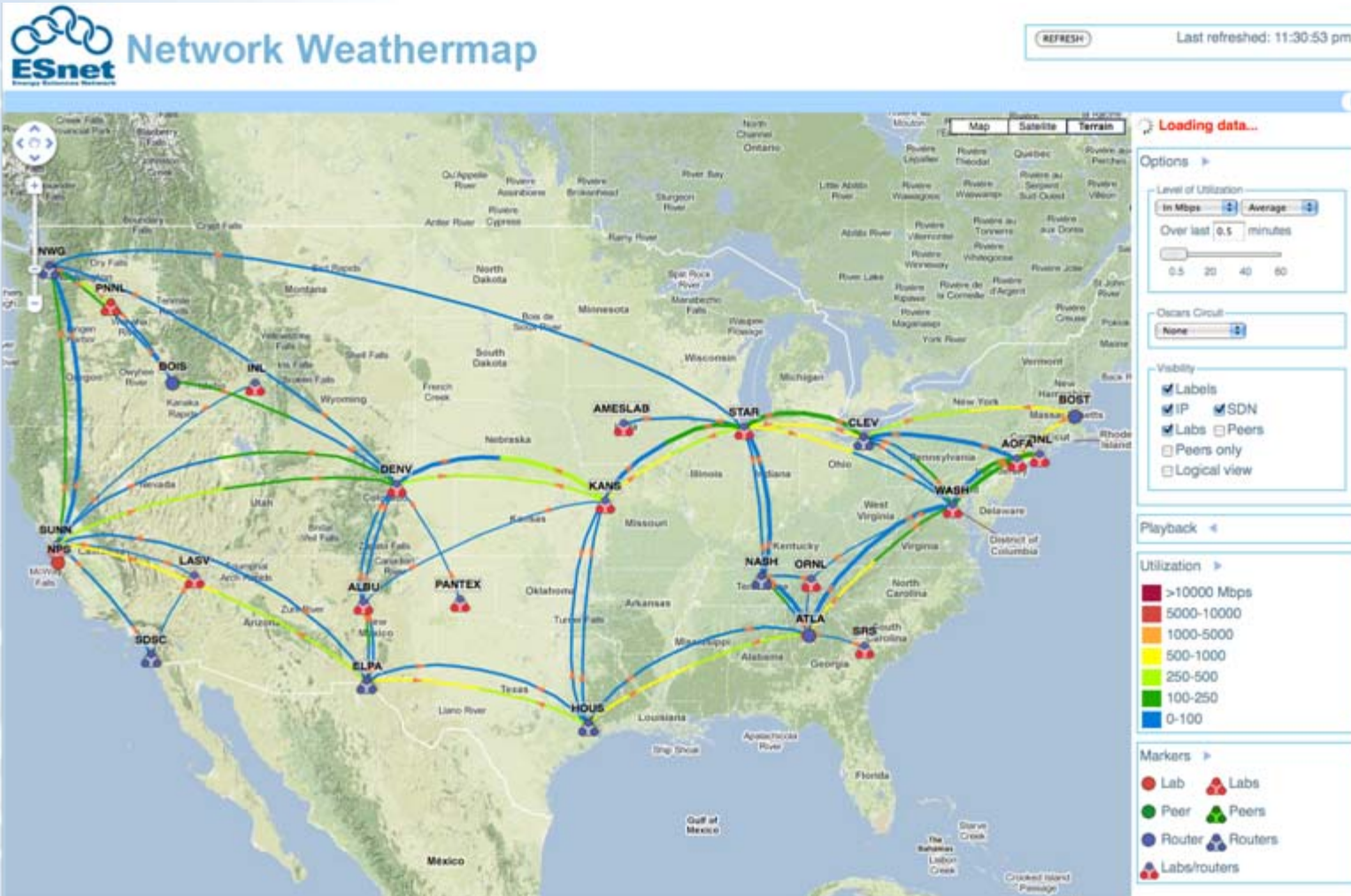
Graphite visualization with Net Almanac annotation



Monthly Stats



Network Weathermap



Supercomputing 2009 and 2010



Primary SNMP data collection system

Source for perfSONAR

Used to judge the bandwidth challenge (2009)

Store and display energy utilization data (2009)



Upcoming Features (Vaporware)

Link upgrade prediction

- Statistical analysis
- Very interested in other people's experiences
- Summer student has initial implementation

Better Caching and I/O Scheduling

Documentation

Official Release



Questions?

ESxSNMP: <http://code.google.com/p/esxsnmp/>

Email: jdugan@es.net

Follow us: <http://esnetupdates.wordpress.com>,
<http://www.twitter.com/esnetupdates>