Centralized Logging for the GLAST Infrastructure Software

Overview

- Perceived requirements
- Technologies surveyed
- NetLogger message format
- Prototype setup
- Database representation
- To-do list

Perceived Requirements / Goals

- Concentration of log output for many processes on many machines to a central location.
- Provision for placing log output into a database (Oracle).
- Polyglot interface
  - C/C++, Java, Python, Perl
- Suitable for either dedicated servers or batch nodes
- Include useful “standard” meta-information in every message automatically
  - Host, user, process id, etc.
- Minimize GLAST-specific coding effort.
- Reasonably self-contained
  - Try to avoid swallowing a big framework (Zope, JBoss, Hibernate/Spring, etc.)

Technologies Surveyed

- **EPICS / CMLOG (http://www.jlab.org/cdev/cmlog.html)**
  - **Existing installed base and expertise at SLAC.**
  - **Customizable message fields.**
  - **Self-contained distribution, but fairly large codebase.**
  - **(Apparently) needs an instance of a UDP server process on the node that's generating the messages to relay to the central collector (not so great for batch nodes).**
  - **Has a built-in database & query capability (not SQL).**
  - **C, C++, and Java interfaces in the distribution.**

Technologies Surveyed (cont'd)

- **JMS (http://java.sun.com/products/jms/)**
  - **Found several examples that tied e.g. log4j into a JMS queue, but mostly "toy" code.**
  - **Appears to require an EJB container or application server like JBoss or WebSphere.**
  - **General-purpose middleware, so must develop the "application layer"**
  - **Additional "machinery" needed to interoperate with C++ / Python / Perl**

Technologies Surveyed (cont'd)

- **NetLogger Toolkit (http://dsd.lbl.gov/NetLogger/)**
  - A component of some Grid tools under development at LBL
  - Defines a simple text format for messages containing key-value pairs
  - Provides C / Java / Python / Perl code for generating & writing formatted messages.
    - Ties in with Log4J and Python Logging module.
  - Provides Python utilities for moving messages around
    - nlforward.py – monitor a directory of message files and send them to another destination as new data is written
    - netlogd.py – receive messages from a socket and write them to a file
  - Python module includes TCP/UDP socket-receiver objects that call a user-specified callback with message data.

NetLogger Message Format

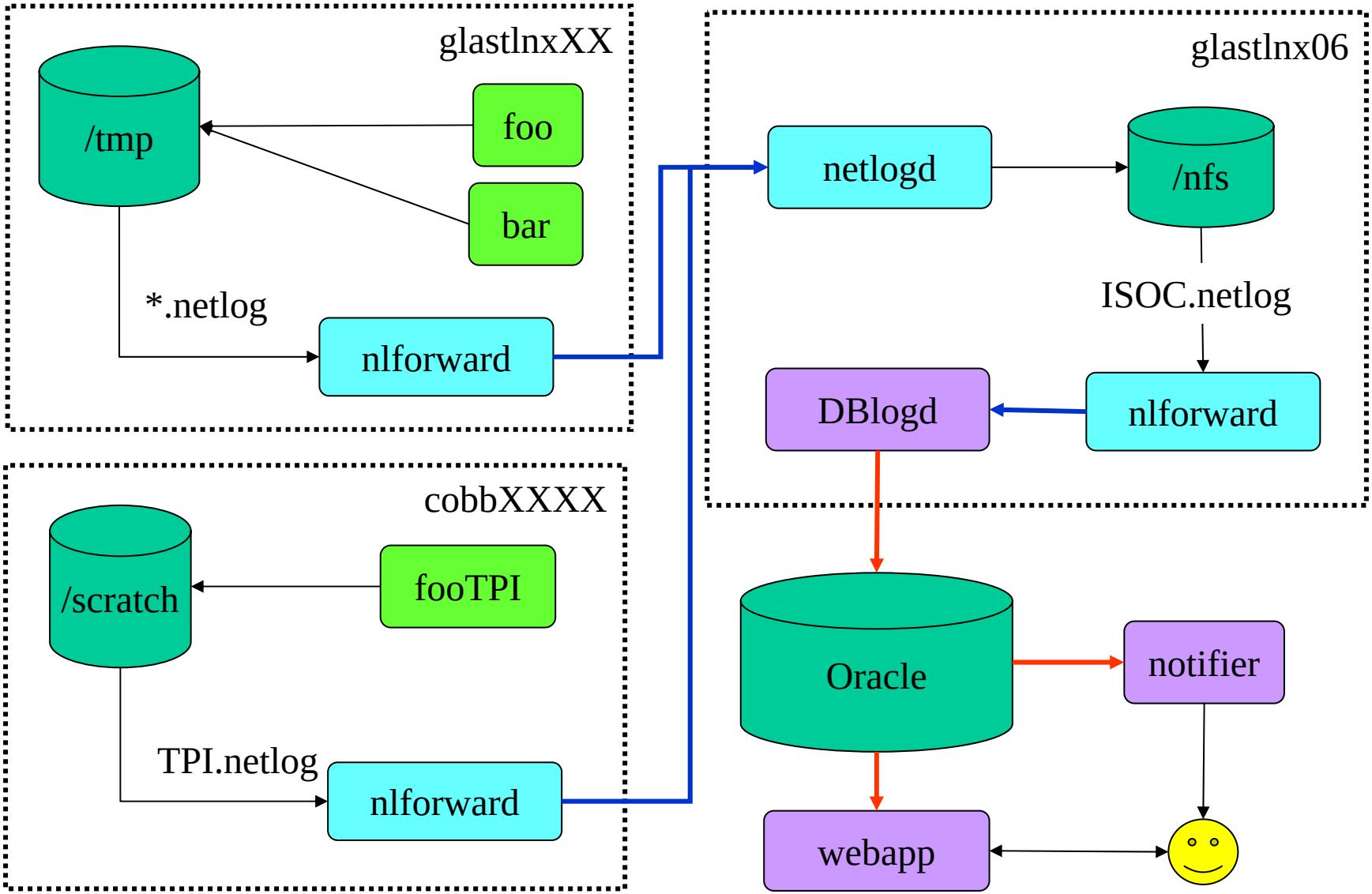
```
t DATE: 2006-06-06T22:05:47.940464
s LVL: INFO
s EVNT: test.evt5
s PROG:
s HOST: 134.79.128.208
s GID: 19a89432f82757ad902bf4c3a9f47837
i PID: 13093
s USER: blee
s MSG: yet another test
s TGT:
```

- **Messages are collections of type-key-value triples**
- **The API provides a way to define “metadata” – fields that appear in every message.**

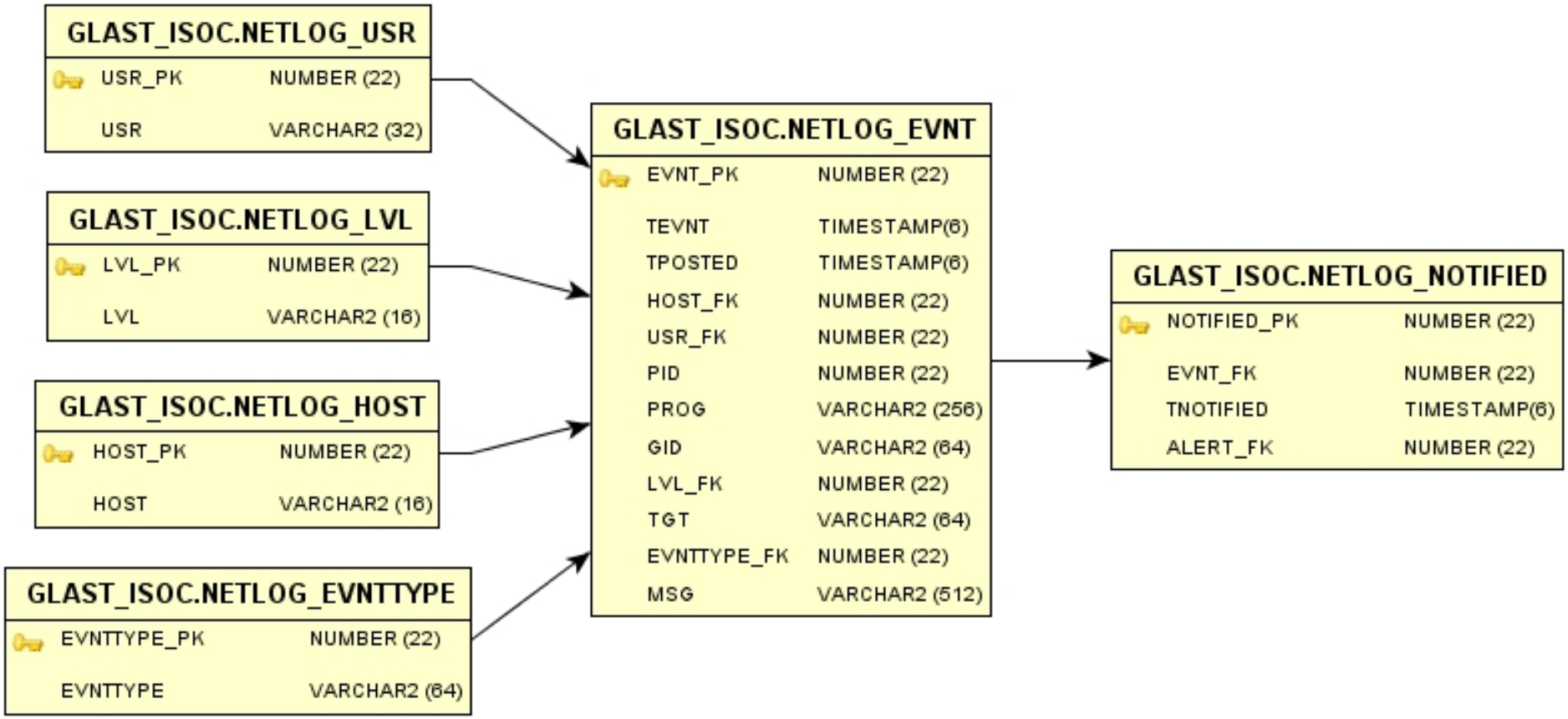
Prototype Setup

- Created init script to run netlogd.py on glastlnx06 and write messages to a file in NFS space.
- Created init script to run nlforward.py on isoc-ops1, glast03, glastlnx06 monitoring /tmp/isoc-netlog-devel and forwarding to x-netlog://glastlnx06:15501.
- Added thin wrapper to CHS core Python code to standardize message content and make a ubiquitous messaging object available.
- Developed simple table-set for database storage of messages.
- Developing a socket-receiver-based application to post messages to Oracle.
- Will use nlforward.py to send messages from NFS central log file to the db-poster application.

Prototype Setup



Database Representation



To-Do List

- Complete db-forwarder application
- Update Logging webapp to use new table structure
- Provide C++ & Java “wrapper” code to enforce message content and local writing location.
- (Eventually) develop alerter application that polls database