

The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing

Rich Wolski, Neil T. Spring, Jim Hayes

Presented by Jik-Soo Kim

Contents

- # Introduction
- # System Architecture
- # Naming and State Management
- # Performance Monitoring
- # Forecasting
- # Reporting Interface
- # Sensor Control
- # Conclusions and Future Work

Introduction

- # Network Weather Service(NWS)
 - Distributed, generalized system for producing short-term performance forecasts based on historical performance measurement
 - Goal
 - Dynamically characterize and forecast the performance deliverable at the application level from a set of network and computational resources

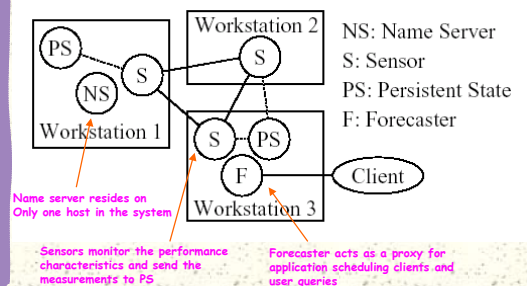
System Architecture(1)

- # Four functional characteristics to be maximized by NWS design
 - Predictive Accuracy
 - Non-intrusiveness
 - Execution longevity
 - Ubiquity

System Architecture(2)

- # Component processes
 - Persistent State process
 - Stores and retrieves measurements from persistent storage
 - Name Server process
 - Implements directory capability used to bind process and data names with low-level contact information
 - Sensor process
 - Gather performance measurements from a resource
 - Forecaster process
 - Produce a predicted value of deliverable performance

System Architecture(3)



Naming and State Management

(1)

- ⌘ All NWS processes are stateless
 - Persistent State is managed explicitly by Persistent State processes
 - System does not maintain any data indefinitely
 - Using circular queue
- ⌘ Naming and Directory service
 - Name-Location Binding
 - Name : human readable text string
 - Location : TCP/IP address & port number

Naming and State Management

(2)

- ⌘ Being Converting the Name Service to use LDAP (Lightweight Directory Access Protocol)
- ⌘ The address of the NWS Name Server process is the only well-known address used by the system
 - Centralized Name Server
 - All other NWS processes register name-location bindings with the Name Server

Performance Monitoring(1)

(1)

- ⌘ There is Tension between the intrusiveness of a monitoring technique and the measurement accuracy it provides
- ⌘ NWS Sensors
 - Gather and store time stamp-performance measurements pairs for a specific resource
 - CPU Sensor
 - Network Sensor

Performance Monitoring(2)

(2)

- ⌘ CPU Sensor
 - Combine information from UNIX system utilities *uptime* and *vmstat* with periodic "active probes" to provide measurements of CPU availability
 - *uptime* & *vmstat* are fairly non-intrusive
 - But, may leave out considerable information
 - Active probes
 - Periodically runs an artificial, compute-intensive "probe" program and calculates the CPU availability as the ratio of its observed CPU occupancy time to the wall-clock time of its execution

Performance Monitoring(3)

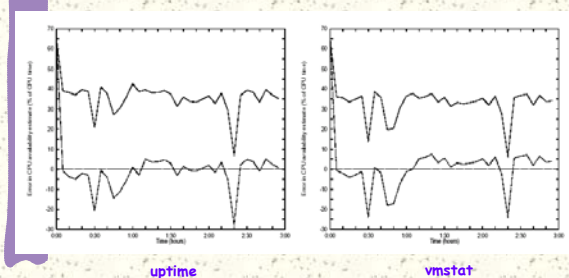
(3)

- ⌘ CPU Sensor
 - Use heuristics to adaptively adjust the frequency with which active probes are conducted => limit the intrusiveness

Performance Monitoring(4)

(4)

Improvement from Active Probing



Performance Monitoring(5)

- ⌘ Network Sensor
 - Rely on active network probes
 - Gathering a set of end-to-end performance measurements from N Sensors require N^2-N probes => Hierarchical Organization of Sensors
 - Measuring network performance characteristics
 - Small-message round-trip time
 - Large-message throughput
 - TCP socket connect-disconnect time

Performance Monitoring(6)

- ⌘ Storing Sensor information to Persistent State
 - To be available to Forecasters
 - The location of the Persistent State Process that a Sensor will use for each of the measurements it gathers is specified when the Sensor is configured
 - When Sensor is initialized
 - Registers the location of the Persistent State Process that stores its measurements data with Name Service

Forecasting(1)

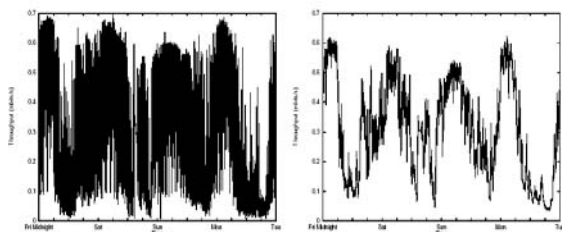
- ⌘ To generate a forecast, a Forecast process requests the relevant measurement history from a Persistent State process
- ⌘ Forecaster works with timestamp-measurement pairs
 - Apply a set of forecasting models
 - Dynamically choose the forecasting technique that has been most accurate

Forecasting(2)

- ⌘ Forecaster process consists of
 - Driver
 - Presents time series to each prediction modules via prediction module interface
 - Keep track of which prediction module yields the lowest aggregate error measure over time
 - Return the forecast predicted by *most accurate* module
 - Set of compile-time determined prediction modules
 - Return a forecast for the next value to driver

Forecasting(3)

⌘ Example Forecasting Results



Reporting Interface

- ⌘ The NWS exports a lightweight and portable C API
 - Contacts the system via sockets
 - Retrieve short term performance forecasts quickly
 - `InitForecaster()`, `RequestForecasts()`
- ⌘ The NWS also provides continuous access to forecasts through WWW
 - CGI Interface
 - Interactive access to Forecasters

Sensor Control(1)

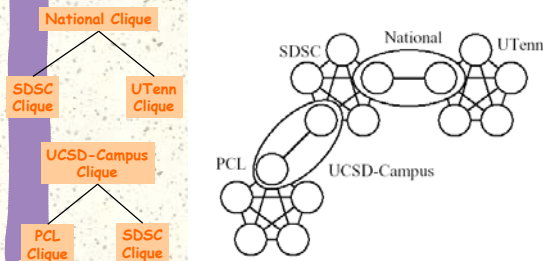
- ⌘ Sensors use adaptive time-out discovery and a distributed leader election protocol to remain stable and limit the load they introduce
- ⌘ Measuring end-to-end network performance is not easy
 - ▣ All-to-all network Sensor communication would consume a considerable amount of resources

Sensor Control(2)

- ⌘ Organizing Sensors as a hierarchy of Sensor sets called *Cliques*
 - ▣ Each Sensor participating in a clique conducts inter-machine experiments with every other clique members but not with Sensors outside the clique
 - ▣ Sensors can participate in multiple cliques
 - ▣ Sensor population may be organized into a hierarchy by defining cliques for each level in the hierarchy and promoting one representative Sensor from each clique to also participate in the clique at the next level

Sensor Control(3)

⌘ Example Clique Organization



Sensor Control(4)

- ⌘ Token passing protocol within Clique
 - ▣ Reduce contention within clique
 - ⇒ Only a single clique member conducts experiments at any given time
 - ▣ Token
 - ▣ right to conduct experiments
 - ▣ Contains ordered list of all Sensors in clique
 - ▣ Leader
 - ▣ Initiate token
 - ▣ Determines the periodicity of re-initiating tokens
 - ⇒ the periodicity of conducting probes

Sensor Control(5)

- ⌘ Token recovery
 - ▣ Leader sets a time-out value for token
 - ▣ Each Sensor calculates a local time-out based on the last time it held the token and the time-out that the leader has determined
 - ▣ When local time-out expires
 - ▣ Assume that token has been lost or the network has been partitioned
 - ▣ Generates a new token
 - ▣ Marks itself as leader

Sensor Control(6)

- ⌘ Managing multiple tokens
 - ▣ Sequencing tokens
 - ▣ Discard old tokens
- ⌘ Adaptive Time-out Discovery
 - ▣ Token time-out value dominates the stability of the token passing protocol
 - ▣ Clique leader use the prediction techniques that are integrated in Forecasters
 - ▣ When a token times out
 - ▣ The time-out value is increased by a fixed amount

Conclusions and Future Work

✦ The implementations of NWS relies on adaptivity to enable

- Stability
- Accuracy
- Non-intrusiveness
- Extensibility

✦ Future Works

- Implementing Name Server using LDAP
- Exploring new forecasting methodologies and performance monitoring facilities