Efficient Data Dissemination through a Storageless Web Database

Thomas Hammel

prepared for DARPA SensIT Workshop

10 October 2000

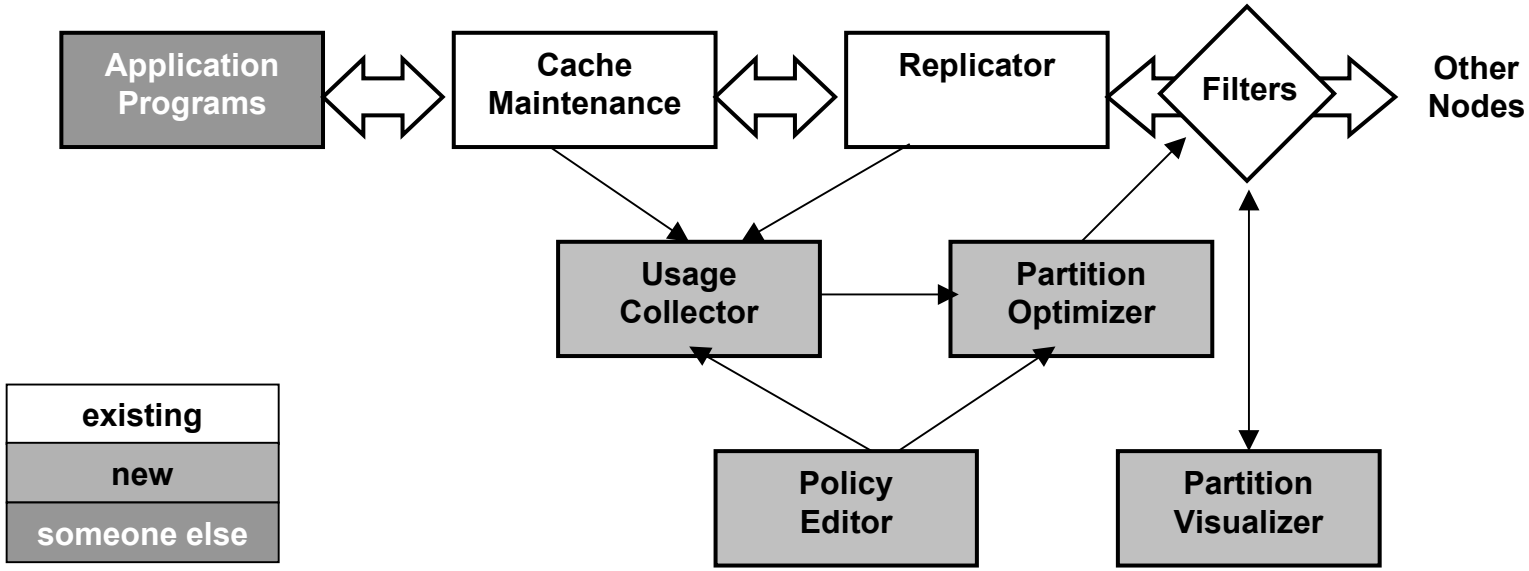
Fantastic Data

207 Prospect Avenue, San Francisco, California 94110, USA • (415) 643-9555 • (415) 643-9554 • www.fandat.com

Web Database System

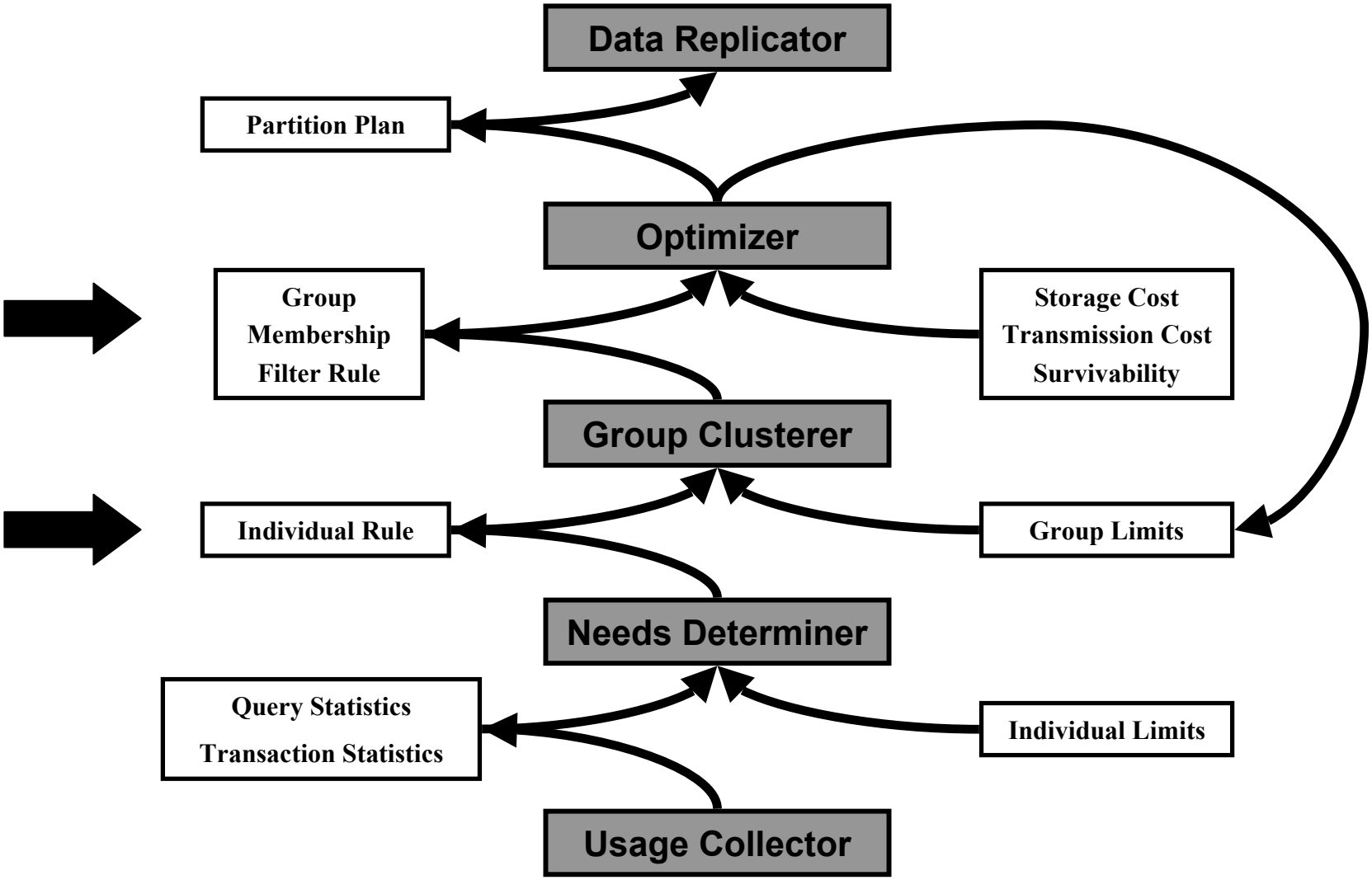
- **Automatically establish redundant data caches throughout the network based on**
  - **data usage patterns, transactions and queries**
  - **optimize cost function based on power consumption, latency, and survivability**
  - **no permanent storage**
- **Disseminate data and maintain redundant caches**
  - **reliable delivery on top of an unreliable channel**
  - **retries mitigated by**
    - **data expiration**
    - **obsolescence detection**
    - **priority**
  - **supports dynamic filter changes**
  - **cooperative repair**

Architecture



- Policy Editor is used to enter data about the organization's policies.
- Usage Collector compiles statistics about the use of the database.
- Partition Optimizer computes and installs the correct filter settings.
- Partition Visualizer provides aids for understanding the system.
- Cache Maintenance maintains the data on an individual node.
- Replicator copies data to other nodes and ensures consistency.
- Application programs do whatever you want.

Partition Determination



Fantastic Data

Dissemination Control Parameters

- **What are they?**
  - **Group (dbgroup in default database)**
    - **group name, address, and port**
    - **What do we use in place of (address, port) for WINS 2.0?**
  - **Membership (dbsupport in default database)**
    - **group, database, node address, yes/no**
    - **What do we use in place of node address for WINS 2.0?**
    - **restriction: a node can't be in different groups for 2 tables that are in the same database**
  - **Rule (dbfilter in specific database)**
    - **group, table, rule (an SQL where clause)**
    - **the dissemination process can execute any rule involving values of fields in the record**

Cache Maintenance and Dissemination Status

- Works under Linux
- API
  - function library complete
  - want to cleanup socket level interface slightly before use
- Replication and cache maintenance servers
  - performs correctly with arbitrary filters
  - caches are currently on disk, need to move to memory
  - uses UDP (broadcast or multicast)
    - How do we interface to WINS 2.0? Diffusion? Radio?
- Remote access
  - currently direct from application to remote server
    - local server provides address of remote server to application
  - want to use local server as intermediary
    - simplifies application
    - allows local server insight into data needs to potentially adjust clustering

Optimizer Status

- **Currently full replication within group**
  - **transmission cost=0**
  - **storage cost=0**
  - **data survivability=1**

  } ➔ **full replication**

- **Partial expected in late 2001**

- **We'll need some configuration data**
  - **node reliability**
  - **transmission cost**
  - **storage limit**

- **And some run-time data**
  - **power**
  - **number of hops to destination**
  - **remaining storage**

Automatic clustering rules

- **Individual rules**
  - **specific value, integer or character**
    - **mode='acoustic'**
    - **type='tank'**
    - **code=347**
  - **range of values (allow 1 side to be unbounded), integer or float**
    - **value>=3 and value<=14**
    - **snr>=3.5**
    - **power>=0.0 and power<=10.0**
  - **area, integer or float**
    - **latitude>=39.342893 and latitude<=39.358214 and longitude>=-120.451740 and longitude<=-120.430266**
- **Combination rules**
  - **"and" and "or"**
    - **mode='acoustic' and snr>3.5 and (latitude>=39.342893 and latitude<=39.358214 and longitude>=-120.451740 and longitude<=-120.430266)**

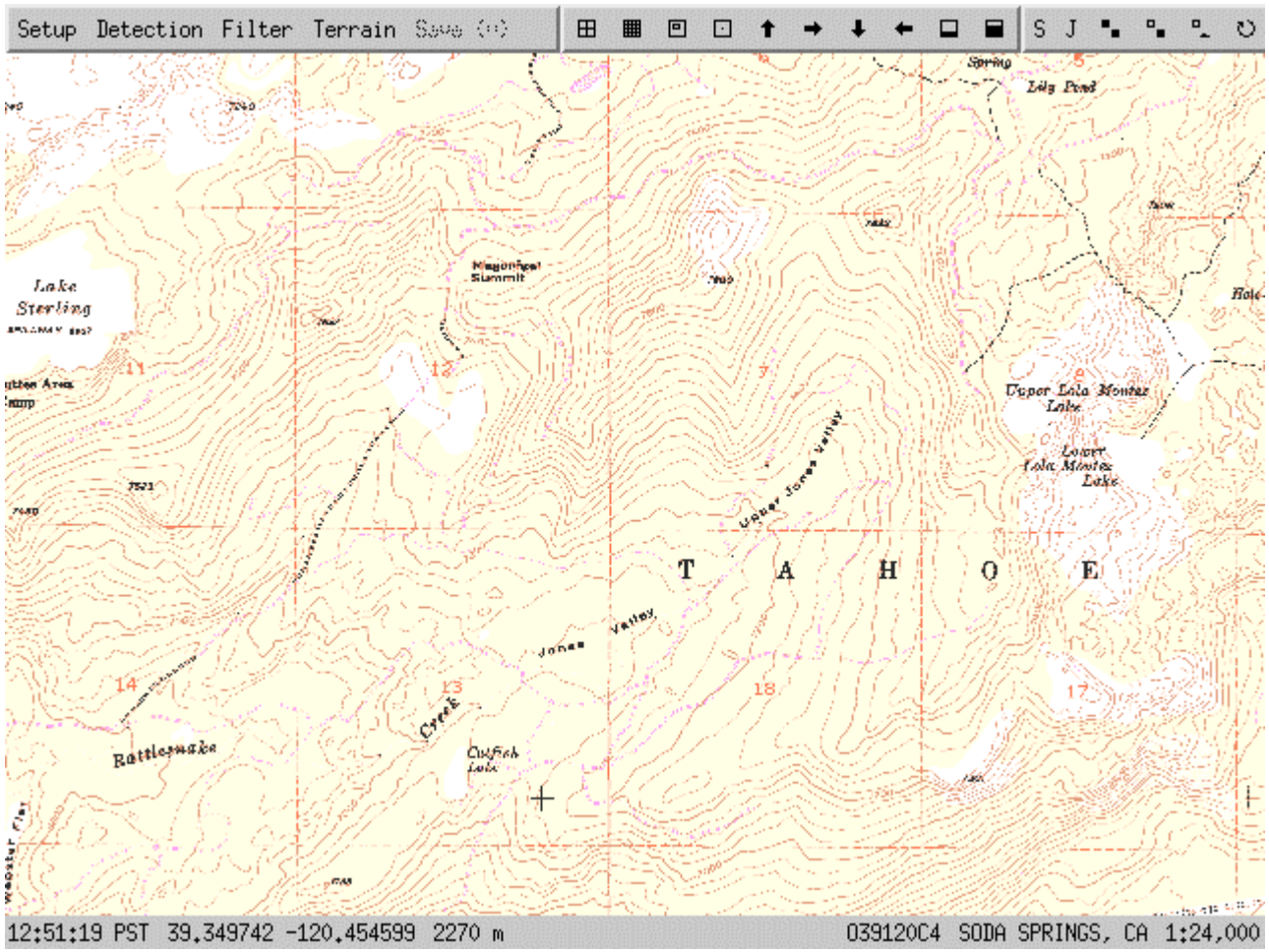
Clustering philosophy

- **locally determined**
  - **not globally optimized**
  - **minimize interaction between nodes required to setup filters**
- **incremental**
  - **try to disturb existing situation as little as possible**
- **filter tolerance**
  - **a little too big, a little too small, that's ok**
- **maintain cluster quality information**
  - **mean coverage of individual needs (percent, record count, bandwidth?)**
  - **excess coverage (percent, record count, bandwidth?)**
  - **number of members in group**
  - **mean age of member's input data (seconds)**

Automatic Clustering Status

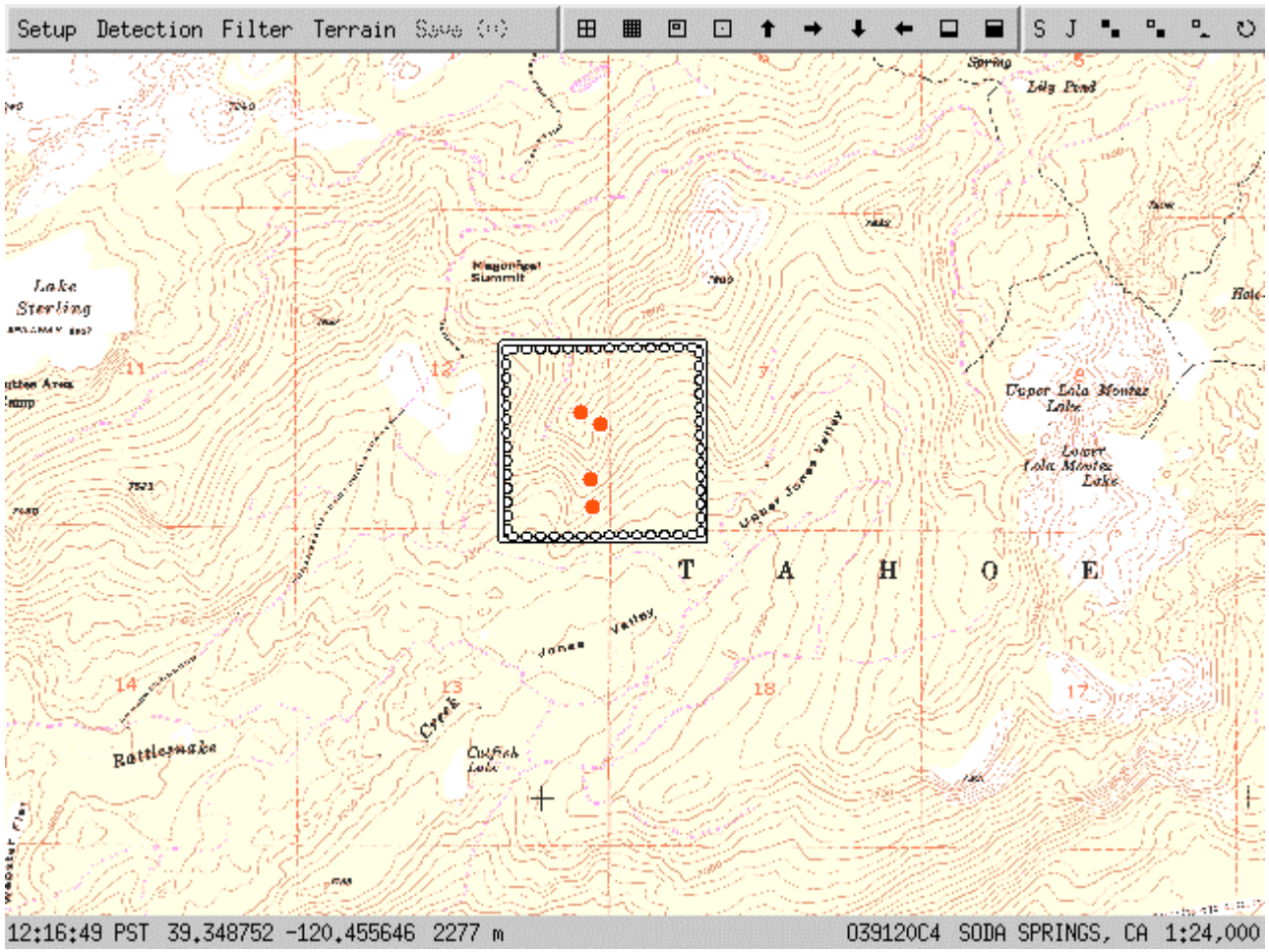
- **Area**
  - **most complicated**
  - **probably also the most useful**
  - **pretty much done**
    - **tune parameters (may make run-time tunable)**
    - **enable shared decision making code (disabled for ease of development)**
    - **test**
- **Specific value**
  - **expected December 2000**
- **Range of values**
  - **simplification of area clusterer**
  - **expected January 2001**
- **Combination**
  - **expected March 2001**

0. Nothing.

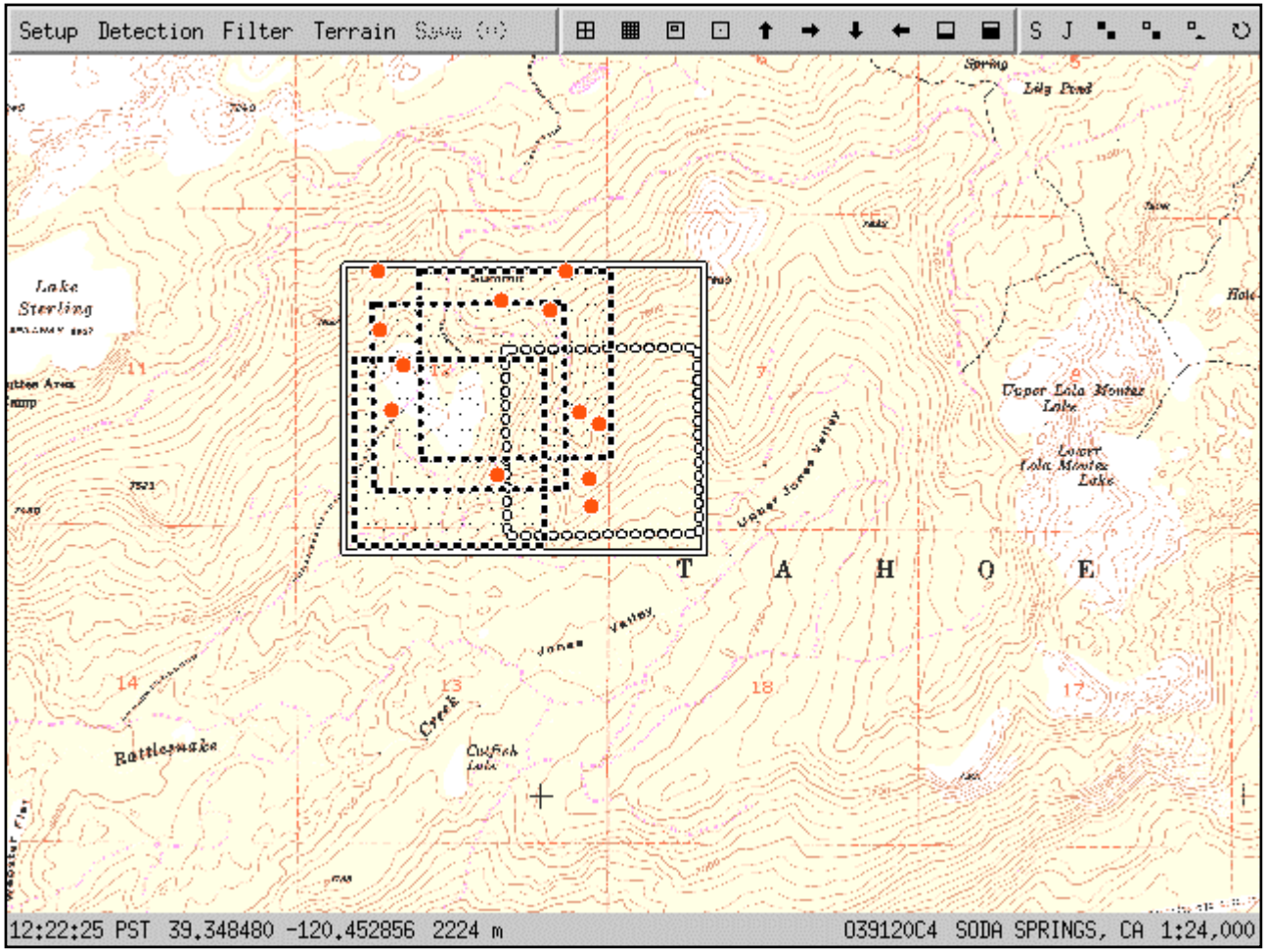


Fantastic Data

1. First sensor. Create new group.



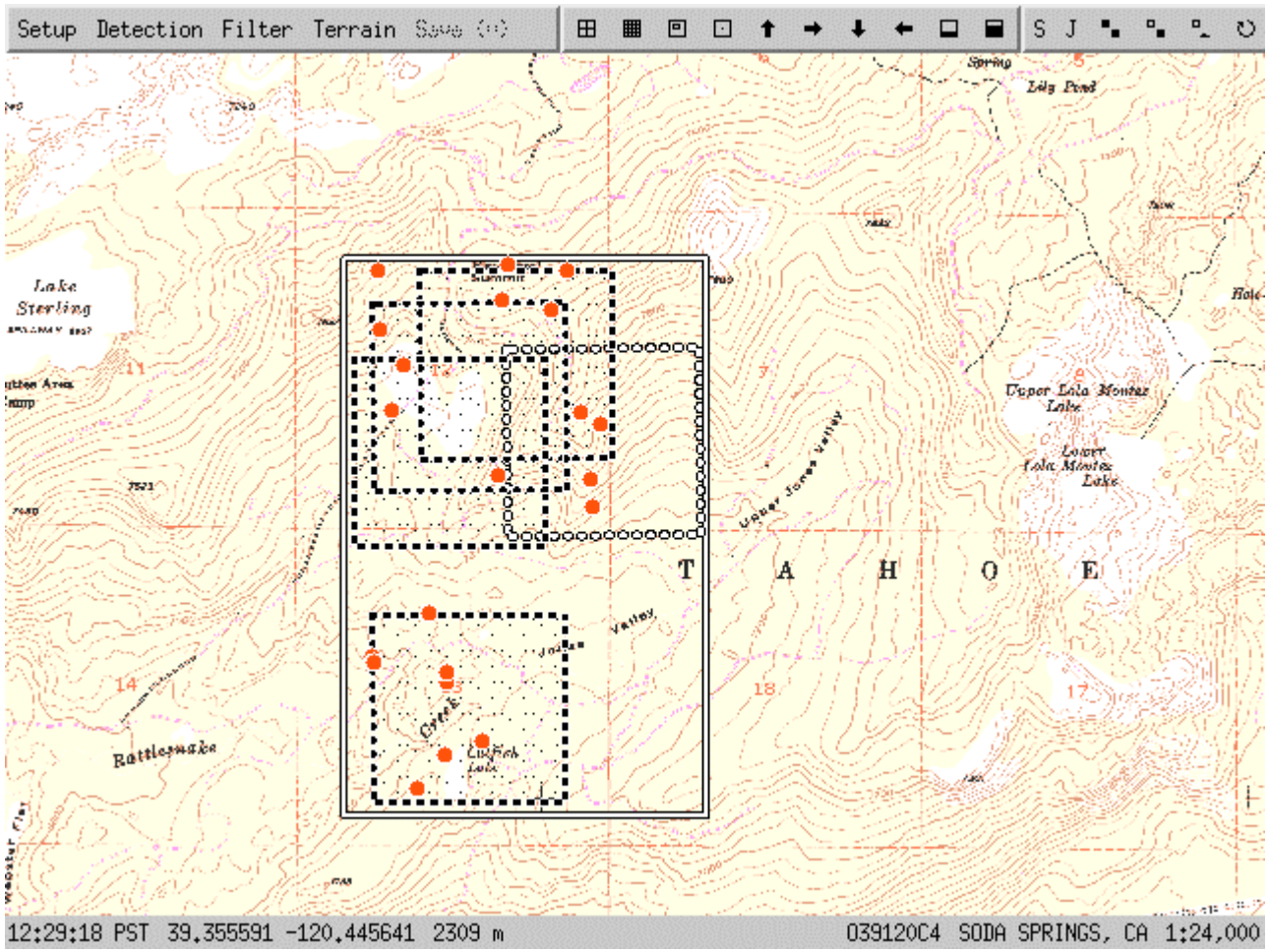
2. Several more related sensors. Extend group.



Fantastic Data

207 Prospect Avenue, San Francisco, California 94110, USA • (415) 643-9555 • (415) 643-9554 • www.fandat.com

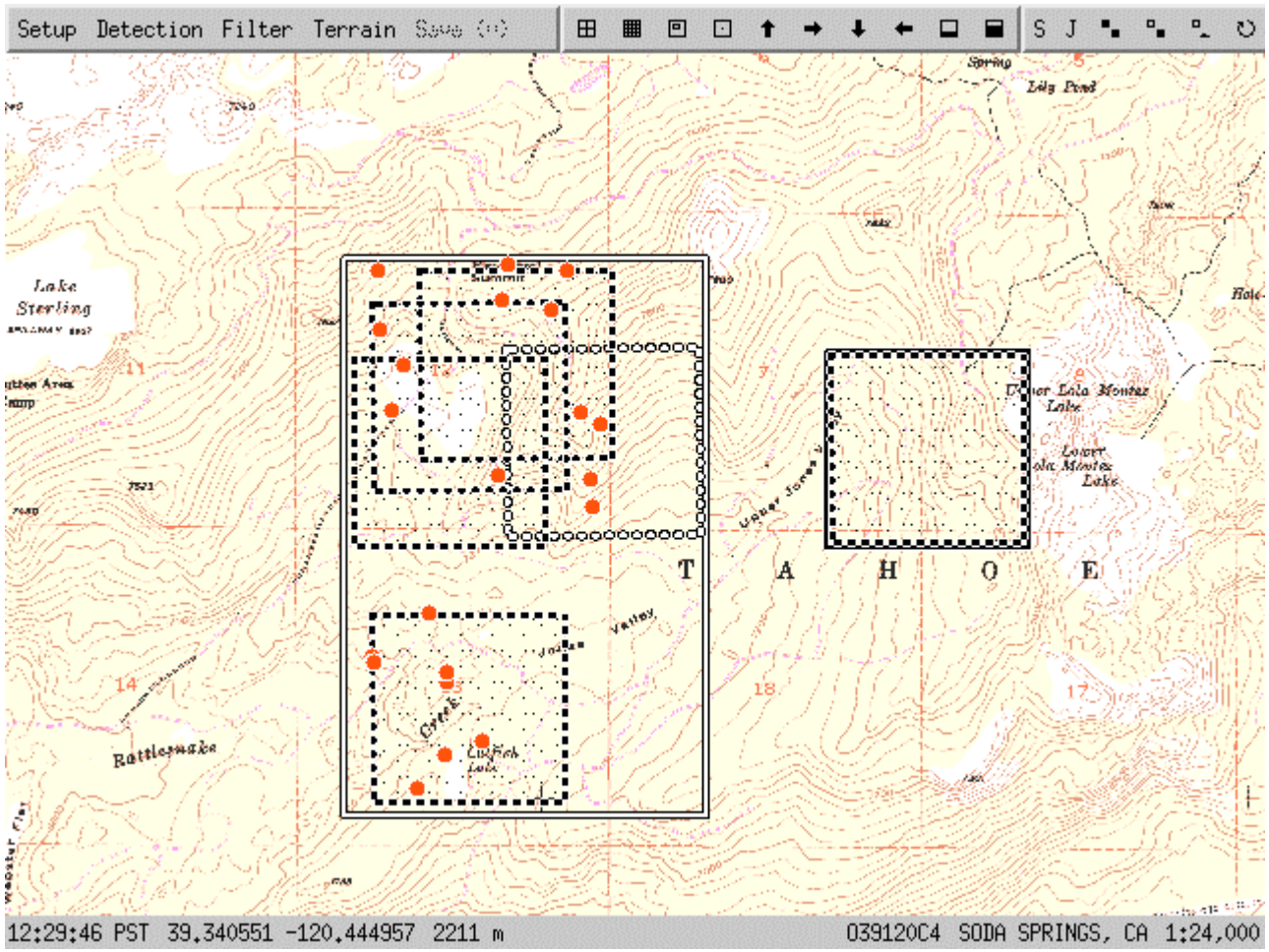
3. Unrelated sensor, but close enough.



Fantastic Data

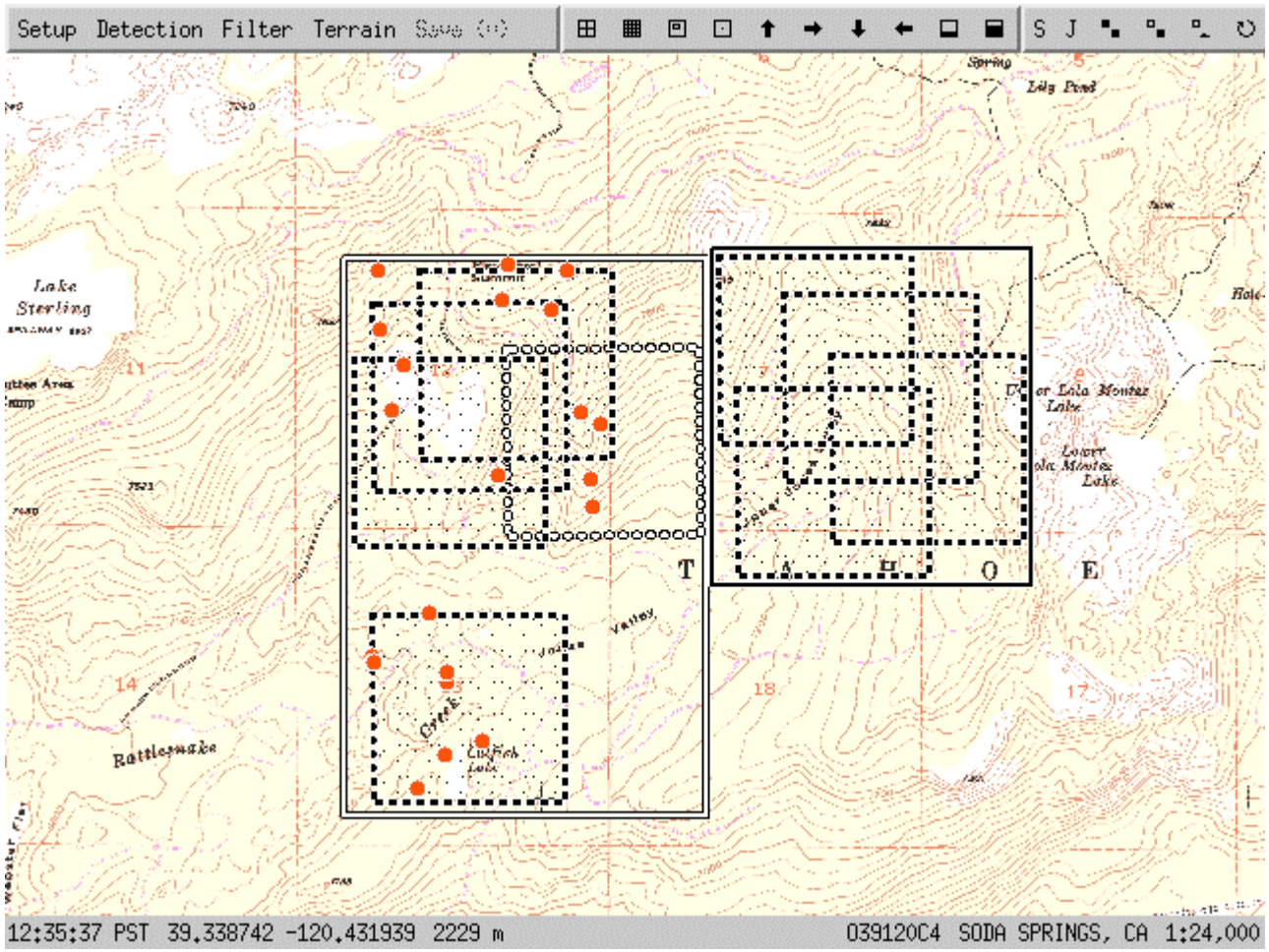
207 Prospect Avenue, San Francisco, California 94110, USA • (415) 643-9555 • (415) 643-9554 • www.fandat.com

4. Unrelated sensor, but too far. Create new group.



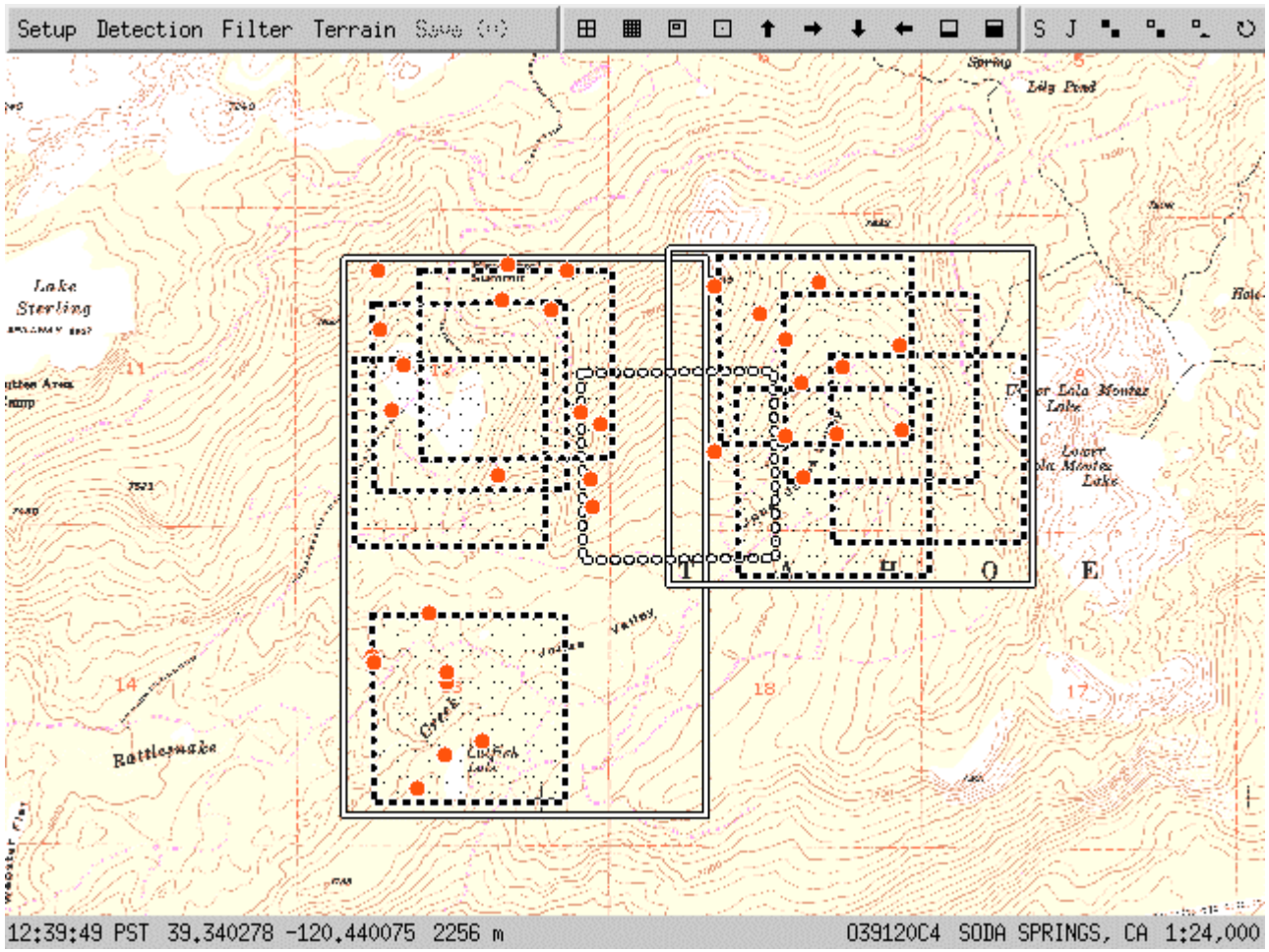
Fantastic Data

5. More sensors. The new group expands.



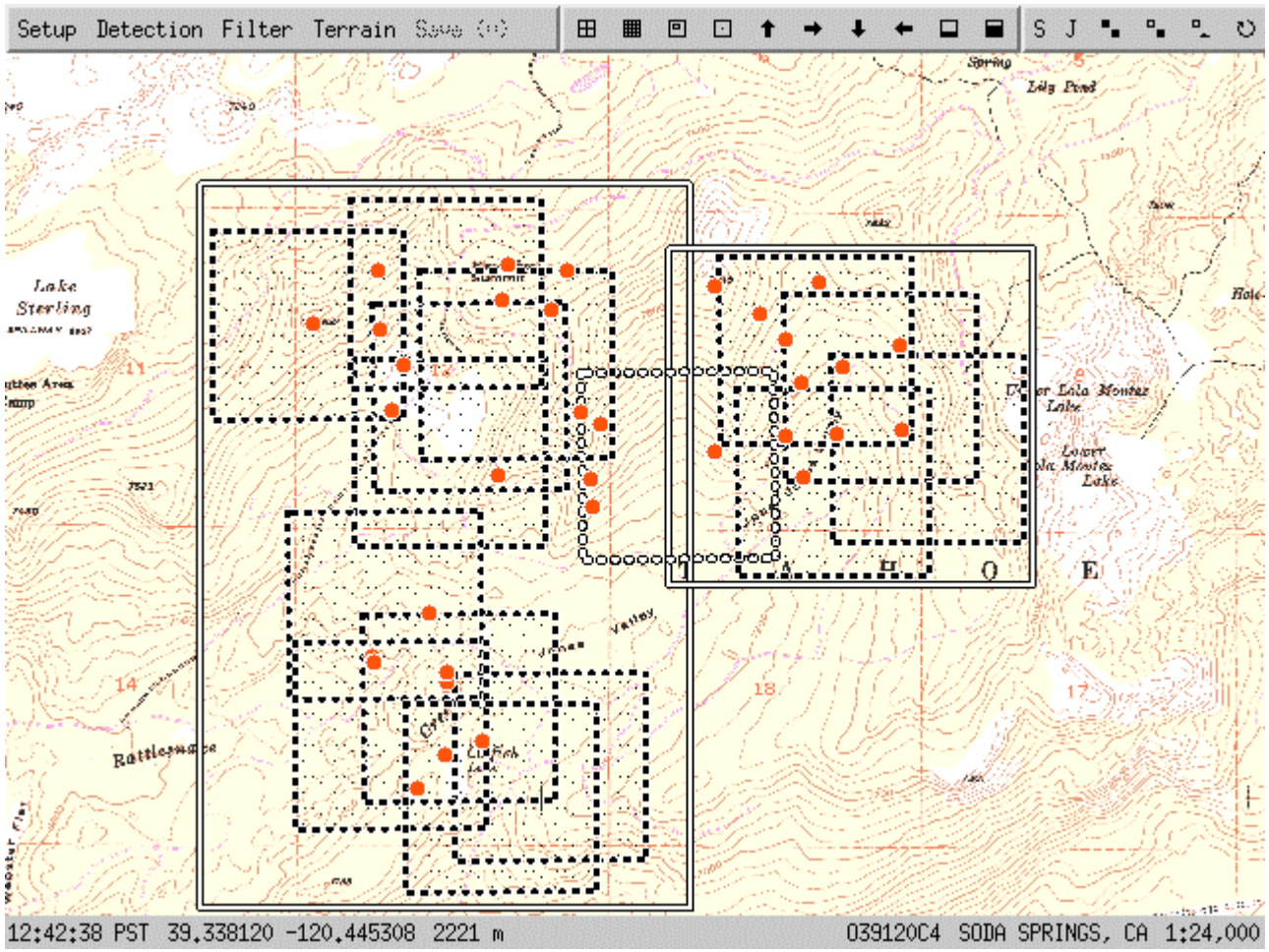
Fantastic Data

6. Sensor joins 2 groups.



Fantastic Data

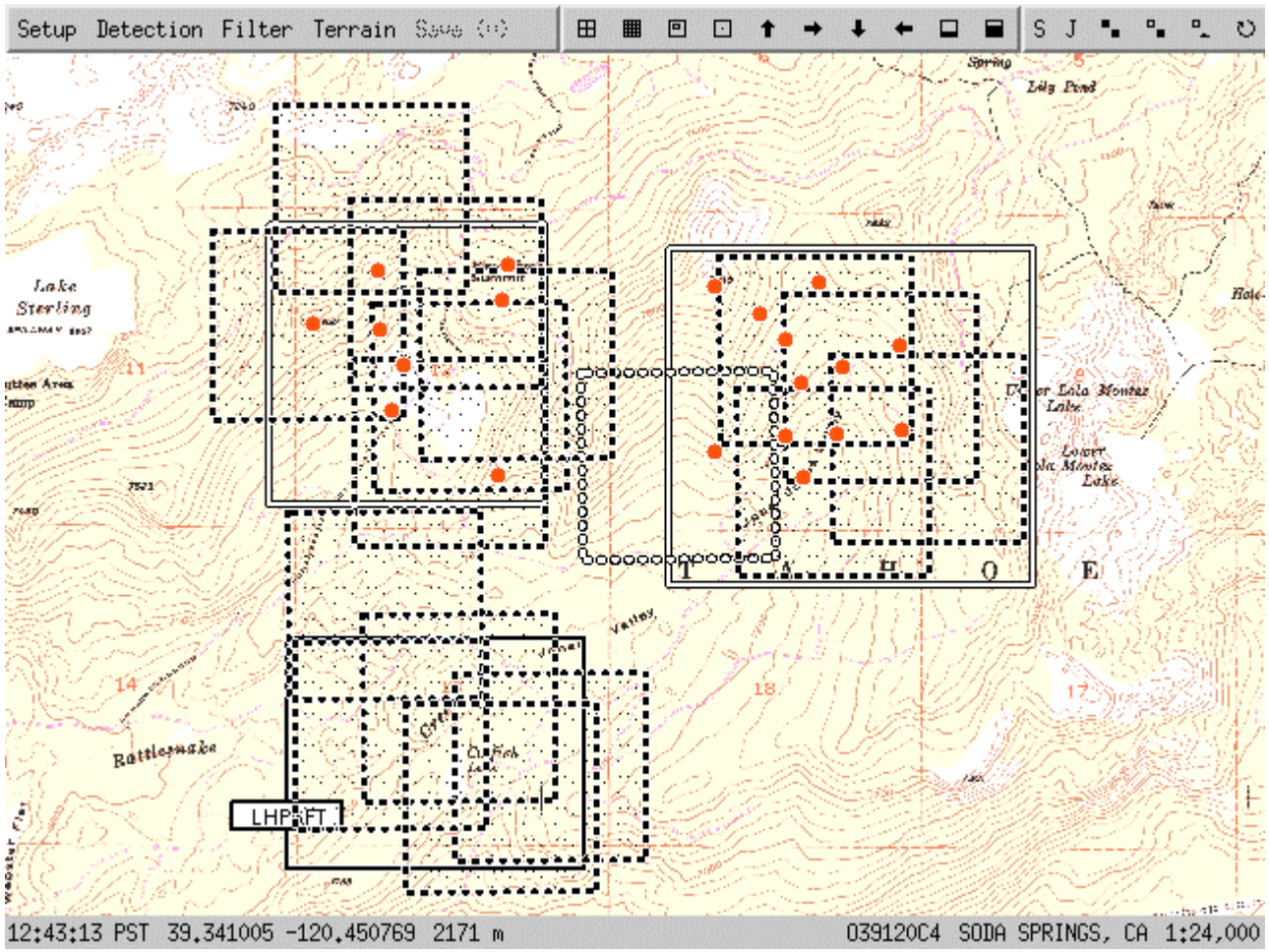
7. Group is expanding close to maximum size.



Fantastic Data

207 Prospect Avenue, San Francisco, California 94110, USA • (415) 643-9555 • (415) 643-9554 • www.fandat.com

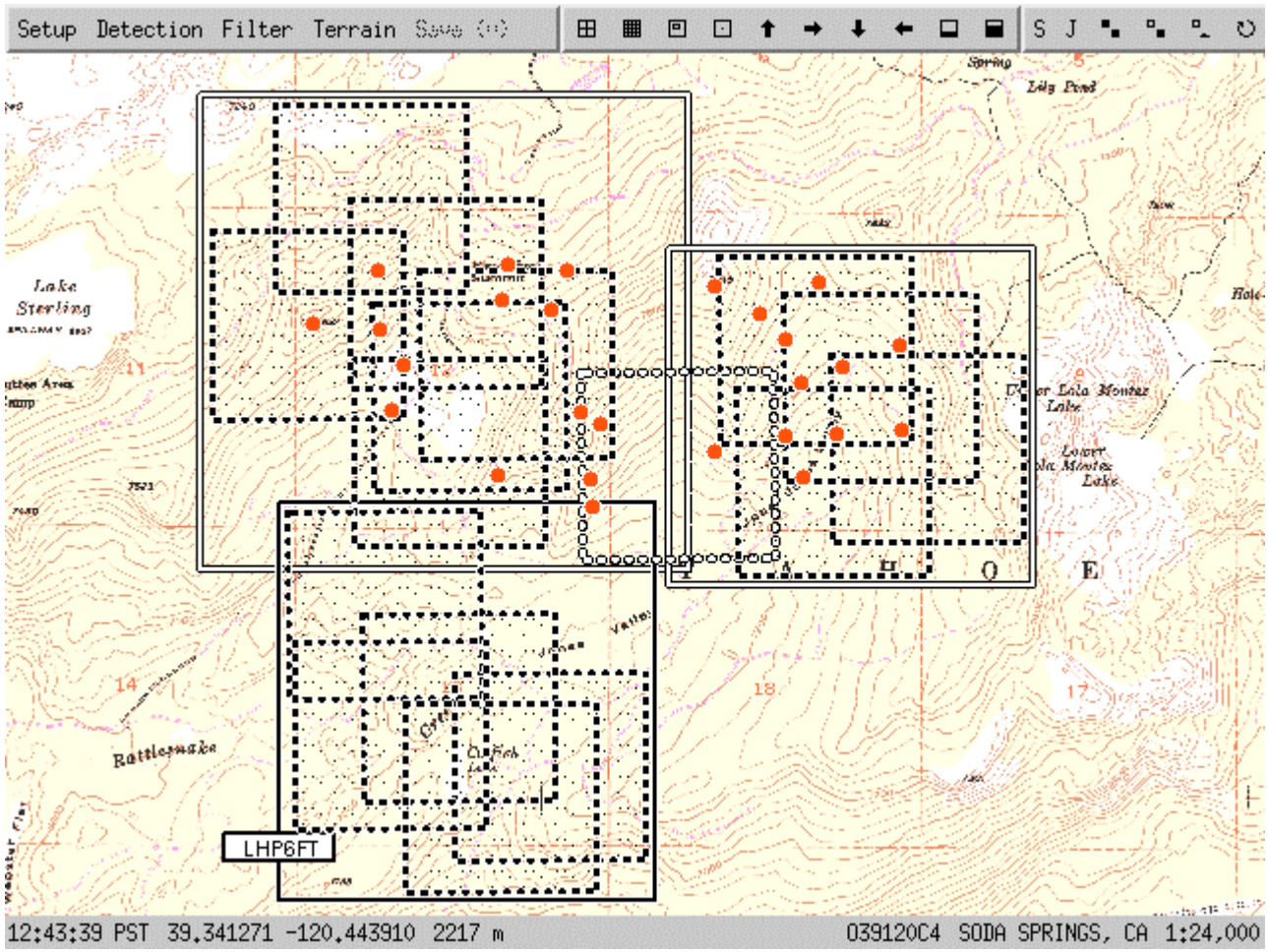
8. Exceeds maximum size. Splits into 2 groups.



This slide represents an intermediate result which usually exists only in the clusterer's memory. It is shown here to illustrate the splitting process.

Fantastic Data

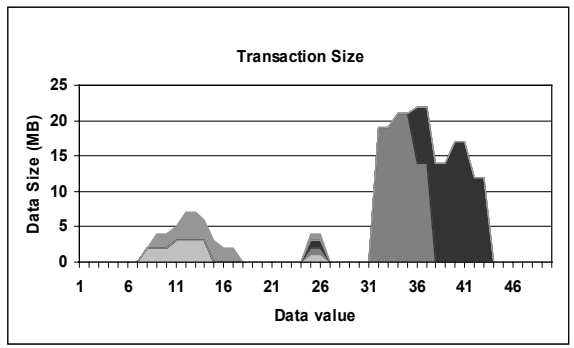
9. After split.



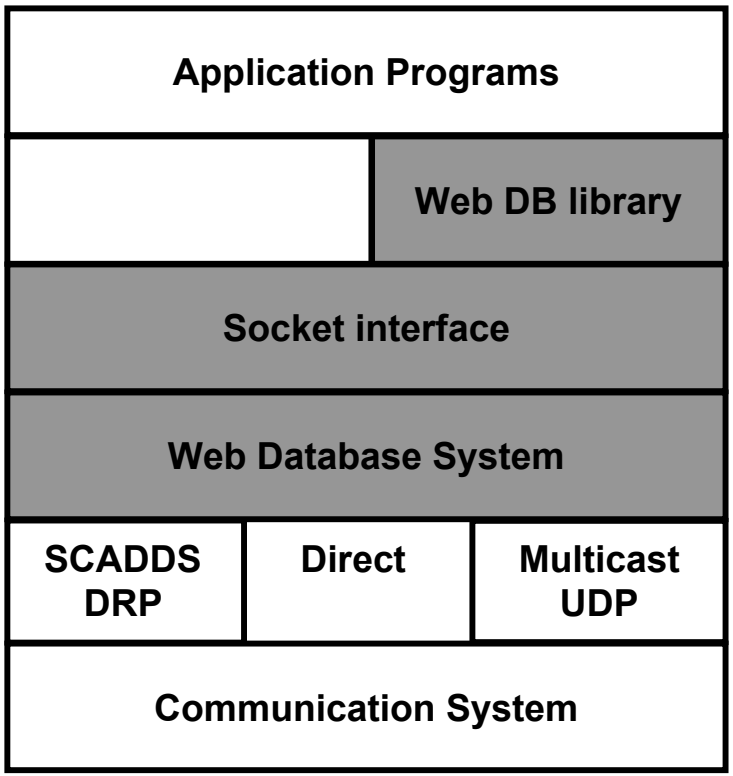
Fantastic Data

Usage Pattern Detection

- Queries (subscription)
  - extract and parse where clause on queries
  - identify fields and conditions that meet one of our 3 standard types
  - need to determine relative weighting of one time, repeated, and persistent queries
    - persistent queries likely to dominate
- Transactions (publishing)
  - Adaptively determine interesting fields over time
  - Build histograms of incoming and outgoing data based on values of interesting fields
  - Form where clause equivalent specifying the regions of interesting data
  - extremely low data rate may require us to configure hints

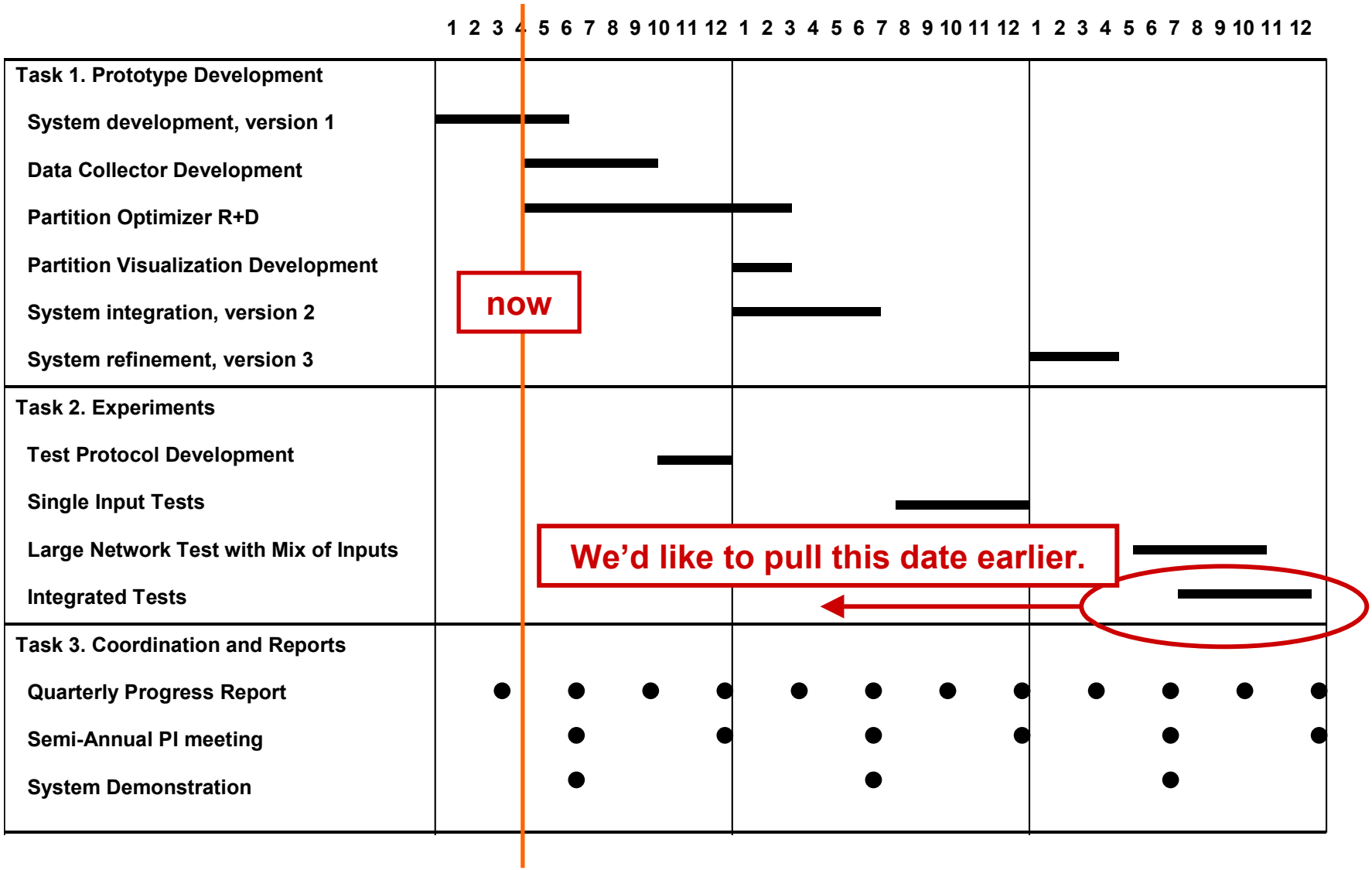


Environment



- **Web Database insulates applications from complexity of shared information space**
  - **access language is SQL**
  - **create table, insert, update, delete, select …**
  - **augmented by put, undelete, notify**
- **Applications may interface by**
  - **including interface library**
  - **by well defined messages delivered over socket**
- **Web database replicators communicate through**
  - **multicast IP**
  - **diffusion layer**
  - **directly to custom radio**
- **Current system available for Linux**

Schedule


1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12
Task 1. Prototype Development
System development, version 1
Data Collector Development
Partition Optimizer R+D
Partition Visualization Development
System integration, version 2
System refinement, version 3
now
Task 2. Experiments
Test Protocol Development
Single Input Tests
Large Network Test with Mix of Inputs
Integrated Tests
We'd like to pull this date earlier.
Task 3. Coordination and Reports
Quarterly Progress Report
Semi-Annual PI meeting
System Demonstration