Report on the NASA Padmini Project

Sandipan Dey,
CSEE Department,
University of Maryland, Baltimore County,
MD, USA,
sandip2@umbc.edu

December 29, 2009

1 The Architecture

1.1 Change in the existing architecture

1.1.1 Separating UI from the Action

UI was tightly coupled with the backend logic in DDMT. Separating the view (panels) from the model/controller (actions) was essential to run DDMT in two different modes: GUI and non-GUI (command-line) mode.

1.1.2 Design of a non-GUI DDMT server

DDMTServer is basically implemented by means of a (multithreaded) server scoket, this is a thick server that runs in the backend, listens to a port and handles various requests by invoking corresponding action handlers.

1.1.3 Design of DDMT WebService

A thin asynchronous Webservice that exposes a list of WebMethods that clients can invoke, it just redirects the client call to the correponding service-handler in the DDMT-Server and uses **RPC**-like mechanisms to communicate with the DDMT server.

1.1.4 Design of Website

The front-end interface from which the users will run distributed algorithms. It communicates with the DDMT Webservice and provides transparancy that is one of the essential features in a distributed system. Html/javascript is used to design the webpage and java applet is used to show graphs.

1.1.5 Design of Database

This DB is created at the webserver to store user-specific information. Jav servlet is used to communicate to the database.

1.1.6 Schematic

Figure - 1 shows the schematic diagram explaining the overall architecture.

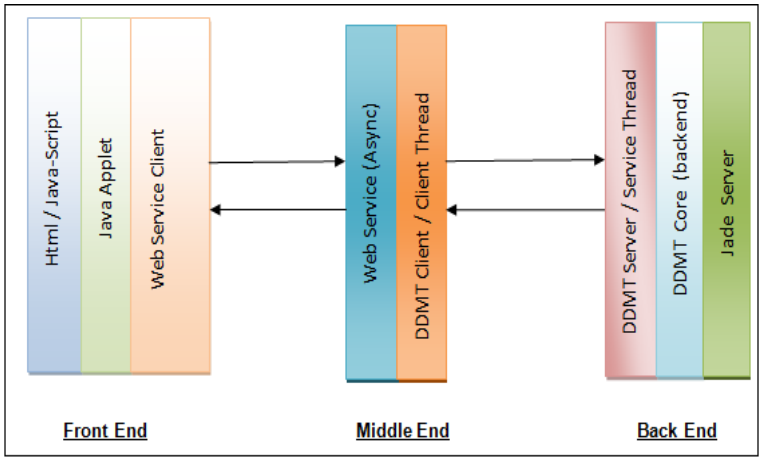

Figure 1: Simplified Multi-tier architecture of DDMT

2 Displaying the Topology of the P2P Network

Java applet is to be used to visualize the topology graph of the nodes in the peer-to-peer network. Adjacency list data structure is to be used to hold the topology information. A sample topology graph is shown in figure - 2.

3 Displaying Dominant 3D EigenVectors for the Eigen-Monitoring algorithm

We have 3-D astronomy data, e.g., the user-chosen attributes of the data are a_g, a_i, a_l from NASA STSS catalog. We run the distributed eigen-state monitoring algorithm [1] on these 3-tuples (do distributed PCA) in order to capture the directions of maximum variance in the feature space and obtain the set of 3 orthonormal eigenvectors, where we choose the two dominant eigenvectors capturing almost all the variance and ignore the 3rd one. Now, since the data tuples for this experiment come from a region of sky, they keep on changing (due to rotation of earth, incoming / outgoing celestial objects etc.) and consequently the eigen vectors for the feature space also keeps on changing

```
50
0 NodeAgent1 None AS_NODE 1 1 3 1.0 13 1.0
1 NodeAgent1 None AS_NODE 1 1 23 1.0
2 NodeAgent1 None AS_NODE 1 1 22 1.0
3 NodeAgent1 None AS_NODE 1 1 0 1.0
4 NodeAgent1 None AS_NODE 1 1 31 1.0
5 NodeAgent1 None AS_NODE 1 1 7 1.0 18 1.0
6 NodeAgent1 None AS_NODE 1 1 14 1.0 25 1.0
7 NodeAgent1 None AS_NODE 1 1 5 1.0
8 NodeAgent1 None AS_NODE 1 1 39 1.0
9 NodeAgent1 None AS_NODE 1 1 42 1.0
10 NodeAgent1 None AS_NODE 1 1 44 1.0
11 NodeAgent1 None AS_NODE 1 1 40 1.0
12 NodeAgent1 None AS_NODE 1 1 36 1.0
13 NodeAgent1 None AS_NODE 1 1 0 1.0 24 1.0 47 1.0
14 NodeAgent1 None AS_NODE 1 1 6 1.0 34 1.0
15 NodeAgent1 None AS_NODE 1 1 20 1.0 24 1.0 30 1.0
16 NodeAgent1 None AS_NODE 1 1 18 1.0 45 1.0
17 NodeAgent1 None AS_NODE 1 1 26 1.0 28 1.0
18 NodeAgent1 None AS_NODE 1 1 5 1.0 16 1.0 35 1.0 44 1.0
19 NodeAgent1 None AS_NODE 1 1 34 1.0 37 1.0
20 NodeAgent1 None AS_NODE 1 1 15 1.0 43 1.0
21 NodeAgent1 None AS_NODE 1 1 29 1.0 45 1.0
22 NodeAgent1 None AS_NODE 1 1 2 1.0 27 1.0
23 NodeAgent1 None AS_NODE 1 1 1 1.0 35 1.0
```

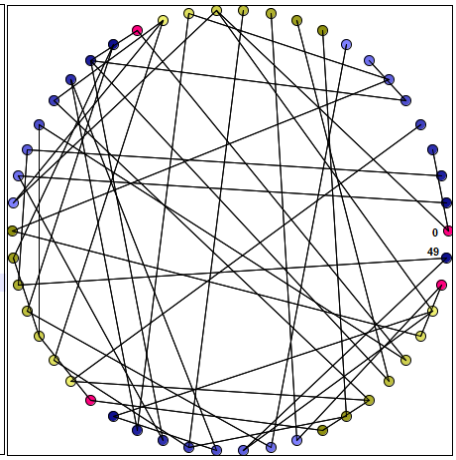

Figure 2: Figure on the left shows a snapshot of the file nodes.txt that contains topology information (as adjacency list), and the figure on the right shows the topology graph applet (for 50 nodes)

(oscillates around a small radius w.r.t. the convergence point). The problem that we address here is to **visualize** the **dominant** (and changing) eigen vectors dynamically (figure - 4).

3.1 Vizualization

The algorithm generates outputs in form of ($\vec{v_1}$,$\vec{v_2}$) tuples where they represent the dominant (orthogonal) eigenvectors, i.e., each output tuple generated by the eigen-monitoring algorithm is of the form $(v_{1x}, v_{1y}, v_{1z}), (v_{2x}, v_{2y}, v_{2z})$.
In graphical representation, each eigenvector will represent a direction in 3D space and moreover since they are orthogonal to each other they will represent two mutually perpendicular directions. It's always good visual representation if we distinguish the two distinct orthogonal eigenvectors (namely 1st most dominant and 2nd most dominant eigen vector) by different color codes. Also, it will be good and easily understandable if we can show how the distributed algorithm convergences to each of the eigenvectors using color codes. We are going to use 3D to 2D mapping technique proposed in [2].
We use properties of **inner product** of two vectors to assign different color codes to the eigen vectors. The following steps describe the technique we used for color code mapping:

- We have two distinct mutually perpendicular directions for the two orthogonal eigen vectors. Now, additionally each vector will have its own radius of convergence that is defined by the respective solid angles $\delta\theta$ or $\delta\phi$, as shown in figure - 3.
- Let the i^{th} row of the output generated by the distributed eigen-monitoring algorithm be denoted by $(\vec{v_1}^{(i)}, \vec{v_2}^{(i)})$. Such output tuples are continuously gener-

ated. We use a backend thread to asynchronously read the stream and update the frontend UI (applet).
Also, let's denote the set of 1st dominant eigen vectors computed by the algorithm by V_1 and set of 2nd dominant eigen vectors by V_2.
Hence, $\vec{v}_1 \in V_1 \wedge \vec{v}_2 \in V_2 \Rightarrow < \vec{v}_1, \vec{v}_2 >= v_1^T.v_2 \approx 0$, by orthogonalaity condition. If additionally, we have orthonormality then
$\vec{v}_1, \vec{v}_1' \in V_1 \Rightarrow < \vec{v}_1, \vec{v}_1' >= v_1^T.v_1' \approx 1$. (Since $< \vec{v}_1, \vec{v}_1 >= 1, < \vec{v}_1', \vec{v}_1' >= 1$, by orthonormality and by **Cauchy-Shwartz inequality** we have
$< \vec{v}_1, \vec{v}_1' > \leq < \vec{v}_1, \vec{v}_1 > . < \vec{v}_1', \vec{v}_1' >= 1$, also since inner product is a similarity measure their product will be close to 1).
Initially, $V_1 = V_2 = \Phi$. Iteratively compute $V_1 = V_1 \cup \{\vec{v_1}^{(i)}\}$ and $V_2 = V_2 \cup \{\vec{v_2}^{(i)}\}$, by adding the corresponding vectors obtained from the i^{th} output touple generated by the algorithm.
At any given instant, we compute the dominant eigenvector directions $\vec{e_1}$ and $\vec{e_2}$ from the already-computed eigen vectors from the set V_1 and V_2 respectively in the following manner (by taking online average):

$\vec{e}_1 = \{e_{1x}, e_{1y}, e_{1z}\}$, with $e_{1x} = \frac{\sum_{v_1 \in V_1} v_{1x}}{|V_1|}$, $e_{1y} = \frac{\sum_{v_1 \in V_1} v_{1y}}{|V_1|}$, $e_{1z} = \frac{\sum_{v_1 \in V_1} v_{1z}}{|V_1|}$.

$\vec{e}_2 = \{e_{2x}, e_{2y}, e_{2z}\}$, with $e_{2x} = \frac{\sum_{v_2 \in V_2} v_{2x}}{|V_2|}$, $e_{2y} = \frac{\sum_{v_2 \in V_2} v_{2y}}{|V_2|}$, $e_{2z} = \frac{\sum_{v_2 \in V_2} v_{2z}}{|V_2|}$.

Let's simplify the above by: $\vec{e}_1 = \frac{1}{|V_1|} \sum_{v_1 \in V_1} \vec{v_1}$ and $\vec{e}_2 = \frac{1}{|V_2|} \sum_{v_2 \in V_2} \vec{v_2}$. As soon as we get a new vector $\vec{v}$ from the backend, we perform the following steps:

- Normalize eigenvector $\vec{v} = \{v_x, v_y, v_z\}$ to obatain $\vec{\vartheta} = \left\{ \frac{v_x}{\sqrt{v_x^2+v_y^2+v_z^2}}, \frac{v_y}{\sqrt{v_x^2+v_y^2+v_z^2}}, \frac{v_z}{\sqrt{v_x^2+v_y^2+v_z^2}} \right\}$.
- Find if $\vec{\vartheta} \in V_1$ or $\vec{\vartheta} \in V_2$.
- Use colors gradients (C_1, C_1') to color the vectors in set V_1 and colors gradients (C_2, C_2') to color the vectors in set V_2.
- Use orhonormality to find set membership, e.g., if $\vec{\vartheta} \in V_1$, the inner product $< \vec{\vartheta}, \vec{e}_1 > \approx 1$ and $< \vec{\vartheta}, \vec{e}_2 > \approx 0$.
- Use gradient colors to represent the dispersion of the eigen vectors in the same set. To find the solid angles, again use the inner products (as similarity measure), e.g., if $\vec{\vartheta} \in V_1$, use the inner product $< \vec{e_1}, \vec{\vartheta} >$ to find how far $\vec{v_1}$ is from the centroid of the set V_1 (i.e., measure the solid angle $\delta\theta$) and accordingly assign a **convex combination** of the color codes C_1 and C_1' to the vector $\vec{\vartheta}$. The vector $\vec{v_1}$ will have more C_1 components in its color if it is closer to $\vec{e_1}$. Consequently it will have more C_1' components if the solid angle between $\vec{e_1}$ and $\vec{v_1}$ is larger. As seen from figure, the eigen vectors closer to $\vec{e_1}$ are yellower, while the ones comparatively away from the convergence are redder.

– Similarly the 2nd dominant eigen vector is colored.

– The algorithms for construction of the sets and color mapping are described in the following section.

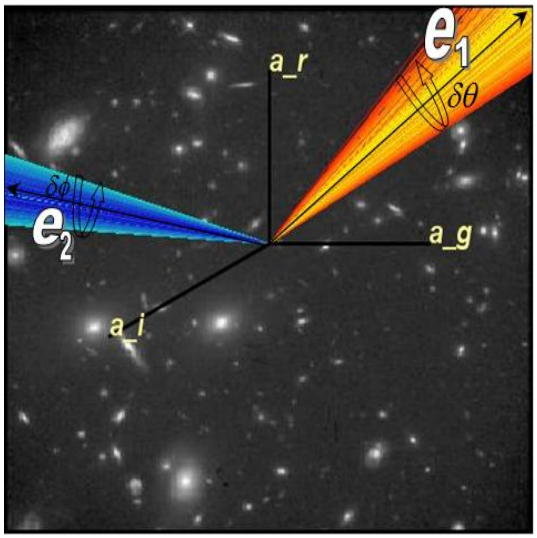

Figure 3: The mapping of color codes in the graph applet: the solid angles of convergence

3.2 Algorithms

Algorithm 1 Construct sets V_1 and V_2

$V_1 \leftarrow \Phi$
$V_2 \leftarrow \Phi$
$\vec{e}_1 \leftarrow 0$
$\vec{e}_2 \leftarrow 0$
for all $(\vec{v_1}^{(i)},\ \vec{v_2}^{(i)})$ **do**
    $\vec{e}_1 \leftarrow \frac{|V_1|.\vec{e}_1+\vec{v_1}^{(i)}}{|V_1|+1}$
    $\vec{e}_2 \leftarrow \frac{|V_2|.\vec{e}_2+\vec{v_2}^{(i)}}{|V_2|+1}$
    $V_1 \leftarrow V_1 \cup \{\vec{v_1}^{(i)}\}$
    $V_2 \leftarrow V_2 \cup \{\vec{v_2}^{(i)}\}$
end for

Algorithm 2 Draw a new (normalized) vector $\vec{\vartheta}$ with proper color

if $< \vec{e_1}, \vec{\vartheta} > \approx 1$ **then** {equivalently $< \vec{e_2}, \vec{\vartheta} > \approx 0$}
  $V_1 \leftarrow V_1 \cup \{\vec{\vartheta}\}$
else if $< \vec{e_2}, \vec{\vartheta} > \approx 1$ **then** {equivalently $< \vec{e_1}, \vec{\vartheta} > \approx 0$}
  $V_2 \leftarrow V_2 \cup \{\vec{\vartheta}\}$
end if
if $\vec{\vartheta} \in V_1$ **then**
  Use $< \vec{e_1}, \vec{\vartheta} >$ to find the solid angle $\delta\theta$ and accordingly assign a convex combination of the colors $\lambda C_1 + (1-\lambda)C_1'$ to $\vec{\vartheta}$, where $0 \leq \lambda \leq 1$, λ being directly proportional to the inner product $< \vec{e_1}, \vec{\vartheta} >$
end if
if $\vec{\vartheta} \in V_2$ **then**
  Use similar logic to color $\vec{\vartheta}$
end if

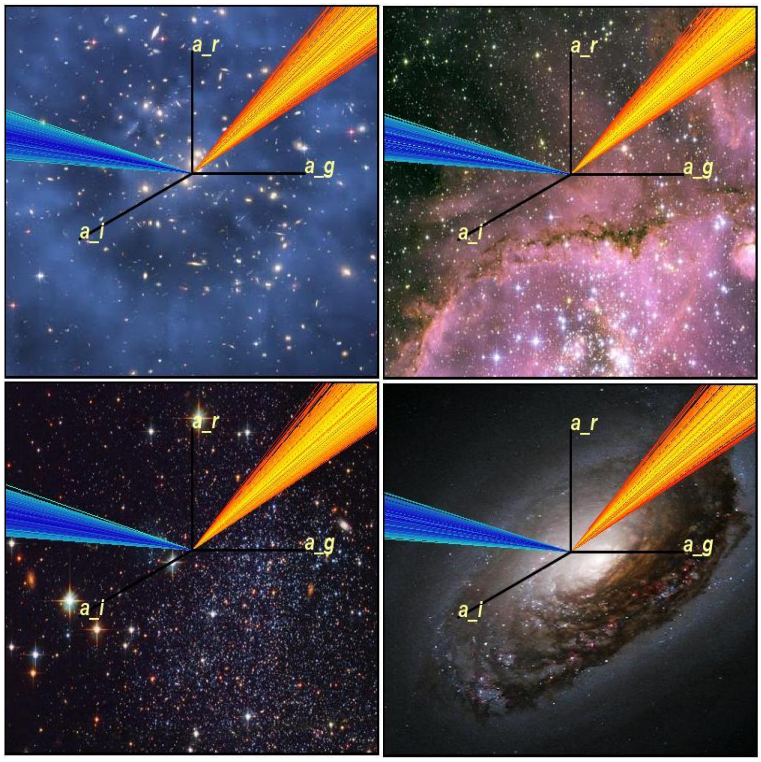

Figure 4: Dominant Eigen Vectors for the Distributed Eigen Monitoring Algorithm

References

[1] Das Kamalika et al. Scalable distributed change detection from astronomy data streams using local, asynchronous eigen monitoring algorithms. 2009.

[2] Dey Sandipan et al. A very simple approach for 3-d to 2-d mapping. 2006.