3rd International Conference on Image and Graphics Dec.18-20, 2004
Hong Kong, China

Adaptive Visualization Pipeline Decomposition and Mapping onto Computer Networks

Mengxia Zhu, Qishi Wu*, Nageswara S. V. Rao*, S. Sitharama Iyengar
Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA
{mzhu, iyengar}@bit.csc.lsu.edu
**Computer Sci. and Math. Div., Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*
{wuqn, raons}@ornl.gov

Abstract

This paper discusses algorithmic and implementation aspects of a remote visualization system, which adaptively decomposes and maps the visualization pipeline onto a wide-area network. Visualization pipeline modules such as filtering, geometry extraction, rendering, and display are dynamically assigned to network nodes to achieve minimal total delay or maximal frame rate. Polynomial-time optimal algorithms using the dynamic programming method to compute the optimal decomposition and mapping are proposed. We implemented an OpenGL-based remote visualization system. We evaluated its performance using a deployment at three geographically distributed nodes.

1. Introduction

A remote visualization system potentially enables an end user equipped with a simple display device and network access to visualize large volumes of scientific data stored and/or rendered at remote sites. Such a system may consist of a remote data source acting as a server, a local display module acting as a client, zero or more intermediate nodes performing operations such as filtering, geometry generation and rendering, and a network connecting all of them together. The performance of such a system critically depends on how efficiently its visualization pipeline is mapped onto the network nodes.

Many existing remote visualization systems employ a predetermined partition of the visualization pipeline and typically send fixed-type data streams such as raw data, geometric primitives, or framebuffer (FB) to remote clients. While such schemes are common, they are not always optimal for high performance visualizations that typically deal with large data sets. Over wide-area connections, this problem is further compounded by the limited bandwidths and time-varying network dynamics. Bowman *et al* [3] proposed a mapping based on predicting the processing times of visualization modules and network bandwidth. Luke *et al* [4] proposed a visualization framework capable of multiple partition scenarios. In these works, the mapping is based on empirical testing and manual configuration.

In this paper, we analytically formulate the problem of optimizing the total delay or frame rate of the visualization pipeline by considering the computation times of the modules and data transfer times between them. Our model highlights the inherent computational aspects of optimally mapping a visualization pipeline onto a network. We propose algorithms using dynamic programming to compute a mapping with minimum total delay or maximum frame rate. The time complexity of these algorithms is $O(n|E|)$, where $|E|$ is the number of edges in the computer network and $n+1$ is the number of visualization modules.

In Section 2, we describe a generic visualization framework. In Section 3, we present our optimal partition and mapping algorithms. Implementation details and test results are provided in Section 4. Conclusions are made in Section 5.

2. Remote visualization system

2.1. Visualization pipeline

Visualization process involves several steps that form the so-called visualization pipeline [1]. Fig. 1 shows a simple visualization pipeline along with the data flow between the pipeline modules. In scientific applications, the raw data is often multivariate and is organized in structures such as NetCDF, and HDF. The filtering module extracts the information of interest from the raw data and performs the necessary preprocessing. The transformation module typically uses a surface fitting technique to derive 3D geometries, or performs shading and classifications for volume rendering. The rendering module converts the transformed geometric or composite volumetric data in 3D view coordinates to a pixel-based image in 2D screen coordinates.

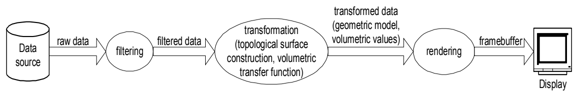
Data source
raw data
filtering
filtered data
transformation (topological surface construction, volumetric transfer function)
transformed data (geometric model, volumetric values)
rendering
framebuffer
Display

Figure 1. A general visualization pipeline.

2.2. Analytical model

The visualization pipeline consists of n+1 sequential modules denoted by $M_1, M_2, \ldots, M_{u-1}, M_u, \ldots, M_{v-1}, \ldots\ldots, M_w, \ldots, M_{x-1}, M_x, \ldots, M_{n+1}$ as shown in Fig. 2. Module M_j, $j = 2, \ldots, n+1$ performs a computational task of complexity c_j on data of size m_{j-1} from module M_{j-1} and generates data of size m_j, which is then sent over the network link to module M_{j+1}. An underlying network consists of k geographically distributed computing nodes denoted by $v_1, v_2, \ldots, v_{k-1}, v_k$. Node $v_i, i = 1, 2, \ldots, k$ has a normalized computing power p_i and is connected to its neighbor node $v_j, j = 1, 2, \ldots, k, j \neq i$ via an edge or link $L_{i,j}$ of bandwidth $b_{i,j}$ and link delay $d_{i,j}$. The transport network is represented by a graph $G = (V, E), |V| = k$, where V and E are sets of nodes and edges, respectively.

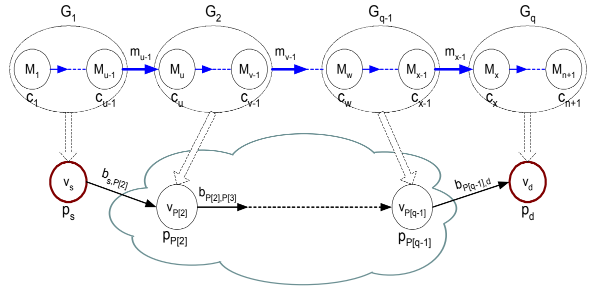

Figure 2. Pipeline partitioning and mapping.

We consider a path P of q nodes from a source node v_s to a destination node v_d in the transport network, where $q \in [2, \min(k, n+1)]$ and path P consists of nodes $v_{P[1]} = v_s, v_{P[2]}, \ldots, v_{P[q-1]}, v_{P[q]} = v_d$. The pipeline is decomposed into q *visualization groups* denoted by $G_1, G_2, \ldots, G_{q-1}, G_q$, which are mapped one-to-one to the nodes of P. The data flow between two adjacent groups originates at the last module in the preceding group such that we have $m(G_1) = m_{u-1}, m(G_2) = m_{v-1}, \ldots, m(G_{q-1}) = m_{x-1}$. The client at last node v_d sends control messages to one or more preceding visualization groups to support interactive operations. However, transport time for control message is assumed to be negligible due to its small size. We consider two optimization problems:

(a) **Minimal total delay**: The goal is to minimize the total time incurred on the forward links from the source node to the destination node, given as follows:

$$T_{total}(Path\ P\ of\ q\ nodes) = T_{computing} + T_{transport}$$
$$= \sum_{i=1}^{q} T_{G_i} + \sum_{i=1}^{q-1} T_{L_{P[i],P[i+1]}}$$
$$= \sum_{i=1}^{q} \left(\frac{1}{p_{P[i]}} \sum_{j \in G_i\ and\ j \geq 2} \left(c_j m_{j-1} \right) \right) + \sum_{i=1}^{q-1} \left(\frac{m(G_i)}{b_{P[i],P[i+1]}} \right) \quad (1)$$

(b) **Maximal frame rate**: Our goal is to maximize the frame rate by minimizing the time incurred on a bottleneck link/node (for applications with streaming data), which is given as follows:

$$T_{bottleneck}(Path\ P\ of\ q\ nodes)$$
$$= \max_{\substack{Path\ P\ of\ q\ nodes \\ i=1,2,\ldots,q-1}} \left\{ T_{computing}(G_i), T_{transport}(L_{P[i],P[i+1]}), T_{computing}(G_q) \right\}$$
$$= \max_{\substack{Path\ P\ of\ q\ nodes \\ i=1,2,\ldots,q-1}} \left\{ \begin{array}{l} \frac{1}{p_{P[i]}} \sum_{j \in G_i\ and\ j \geq 2} \left(c_j m_{j-1} \right), \\ \frac{m(G_i)}{b_{P[i],P[i+1]}}, \\ \frac{1}{p_{P[q]}} \sum_{j \in G_q\ and\ j \geq 2} \left(c_j m_{j-1} \right) \end{array} \right\} \quad (2)$$

In Eqs (1) and (2), we assume that the transport time between modules within a group assigned to a single node is negligible. When $q = 2$, this system reduces to the conventional client and server structure. A special case of this problem where the network nodes form a linear arrangement is considered in [6].

3. Mapping for remote visualization system

3.1. Bandwidth measurement

Three main types of delays contribute to the total message delay, namely, link propagation delay d_p imposed at the physical layer level, equipment-related delay d_q mostly incurred by processing and buffering at the hosts and routers, and bandwidth-constrained delay d_{BW}. We measure the end-to-end delay in transmitting a message of size *r* on a path *P* with *l* links as follows:

$$d(P,r)=d_{BW}(P,r)+\sum_{i=1}^{l}(d_{p,i}(P)+d_{q,i}(P,r)) \quad (3)$$

For large data transfers only the first term of Eq (3) is significant which leads to the linear approximation: $d(P,r)\approx r/EPB(P)$. Here *EPB* denotes the *Effective Path Bandwidth* and is estimated via linear regression using the active measurement technique of [2].

3.2. Partition and mapping

A general partition and mapping problem is similar to the classical graph clustering problem, which is NP-complete [5]. By exploiting the linear arrangement of the visualization modules, we develop polynomial-time algorithms for the problems formulated in Section 2.2.

3.2.1. Minimal total delay. We consider two versions of the total delay minimization problem. The first one considers a one-to-one mapping from visualization modules to network nodes, and the second one combines modules into groups. For the first case, let $T^k(v_i)$ denote the minimal delay with *k* hops from v_s to v_i, which satisfies the following recursion:

$$\underset{k=1\,to\,n,v_i\in V}{T^k(v_i)}=\left\{\min_{u\in adj(v_i)}\left(T^{k-1}(u)+{}^{c_k m_k}\!/_{p_{v_i}}+{}^{m_k}\!/_{b_{u,v_i}}\right)\right\} \quad (4)$$

This recursion follows from the observation that the minimal delay to v_i with *k* hops is the minimum of the delays to its neighbor with *k*-1 hops plus the cost incurred by that link. The base conditions are computed as:

$$\underset{v_i\in V,and\neq s}{T^1(v_i)}=\begin{cases}{}^{c_1 m_1}\!/_{p_{v_i}}+{}^{m_1}\!/_{b_{s,v_i}}, & \forall e_{s,v_i}\in E\\ \infty, & otherwise\end{cases}.$$

The complexity of this algorithm is $O(n\times|E|)$.

For second case, let $T^m(v_i)$ denote the minimal total delay with the first *m* messages mapped onto the network from source node v_s to end node v_i. Then, $T^m(v_i)$ can be computed recursively as follows:

$$\underset{m=1\,to\,n,v_i\in V}{T^m(v_i)}=\min\left\{\begin{array}{l}T^{m-1}(v_i)+{}^{c_m m_m}\!/_{p_{v_i}},\\ \min\limits_{u\in adj(v_i)}\left(T^{m-1}(u)+{}^{c_m m_m}\!/_{p_{v_i}}+{}^{m_m}\!/_{b_{u,v_i}}\right)\end{array}\right\} \quad (5)$$

with the base conditions computed as:

$$\underset{v_i\in V,and\neq s}{T^1(v_i)}=\begin{cases}{}^{c_1 m_1}\!/_{p_{v_i}}+{}^{m_1}\!/_{b_{s,v_i}}, & \forall e_{s,v_i}\in E\\ \infty, & otherwise\end{cases} \text{ and}$$

$$T^m(v_s)=\sum_{i=1}^{m}\left({}^{c_i m_i}\!/_{p_s}\right).$$

In Eq (5), $T(i,j)$ computes the minimal of the two following scenarios based on adding the module M_{m+1} to the partial pipeline. In scenario 1, we execute M_{m+1} at node v_i itself, and add its computing time to $T^{m-1}(v_i)$, a sub-problem of v_i of size *m*-1. In scenario 2, we map link of M_{m+1} to a network link from among all links incident to node v_i, and choose the minimum as in the second term of Eq (5). Thus in each iteration $T^m(v_i)$ either inherits the mapping scheme from $T^{m-1}(v_i)$ by simply adding module M_{m+1} to the last group, or just starts a separate group with module M_{m+1} to the mapping scheme of $T^{m-1}(u),u\in adj(v_i)$. The complexity of this algorithm is $O(n\times|E|)$.

3.2.2. Maximal frame rate. For animation and monitoring tasks, data is continuously generated, manipulated, and rendered. The maximal frame rate that a pipelining can achieve is decided by the slowest transport link or computing node. A modified dynamic programming method of previous section solves this problem. Let $1/F^m(v_i)$ denote the maximal frame rate with the first *m* messages mapped onto the network from source node v_s and the end node v_i. Let $GS^m(v_i)$ represent the sum of message sizes of modules on v_i with the first *m* messages mapped from v_s to v_i. We have the following recursion:

$$\underset{m=1\,to\,n,v_i\in V}{F^m(v_i)}=\min\left\{\begin{array}{l}\max\left(F^{m-1}(v_i),{}^{\left(GS^{m-1}(v_i)+c_m m_m\right)}\!/_{p_{v_i}}\right),\\ \min\limits_{u\in adj(v_i)}\left(\max\left(F^{m-1}(u),{}^{c_m m_m}\!/_{p_{v_i}},{}^{m_m}\!/_{b_{u,v_i}}\right)\right)\end{array}\right\} \quad (6)$$

with the base conditions computed as:

$$\underset{v_i\in V,and\neq s}{F^1(v_i)}=\begin{cases}\max\left({}^{c_1 m_1}\!/_{p_{v_i}},{}^{m_1}\!/_{b_{s,v_i}}\right), & \forall e_{s,v_i}\in E\\ \infty, & otherwise\end{cases} \text{ and}$$

$$F^m(v_s)=\sum_{i=1}^{m}\left({}^{c_i m_i}\!/_{p_s}\right).$$

Here, the bottleneck for each possible scheme is computed, and the one with the minimal time is chosen, which will achieve maximal frame rate.

4. Implementation and case study

Our remote visualization system is deployed at three nodes located at North Carolina State University (NCSU), Oak Ridge National Laboratory (ORNL), and Louisiana State University (LSU). The parallel isosurface extraction computation is implemented on Orbitty cluster at NCSU which consists of 23 nodes (total of 92 CPUs each at 2.4GHz, and total flops of 441.6G). Linux workstations with 3GHz CPU are used as hosts at ORNL and LSU. Our system provides functionalities of scalar glyphs, vector glyphs, isosurface, ray casting using Fastvox 1.0 and animation. Data transmission is carried out via TCP sockets. In this initial implementation, our system runs in a client/server mode without intermediate nodes. The decomposition is optimized (albeit among only two network nodes) since the data sizes exchanged between them varies depending on the grouping. The server estimates the delay time based on the entity being visualized and the available bandwidth, and designates proper visualization modules to the clients. Table 1 illustrates the estimated transport time between LSU and ORNL with different types of data transmitted. The message sizes for raw data, 3D geometry and FB are estimated at the server. The estimated transport delay is calculated as: $T_{delay} = d + (Msg\_size \times 8) / EPB$.

Table 1. Horizontal split test.

Dim	Est. BW (Mbps)	Min delay (sec)	Raw data size/delay	Geometry size/delay	FB size/delay
Case 1: 10x6x8	0.284	0.032	8 K / 0.257sec	1K / 0.032sec	1.8M/50.73sec
Case 2: 50x20x39	0.300	0.034	610K / 16.3sec	16K / 0.46sec	1.8M/48.03sec
Case 3: 150x210x139	0.277	0.033	71.6M / 34.4min	2.4M / 69.34sec	1.8M/52.01sec
Case 4: 256x256x80	0.239	0.033	81.9M / 45.69min	NA	1.8M/60.28sec

Case 1: Cube 1 has a tiny size of geometry and raw data, and hence either can be sent in less than a second.
Case 2: Cube 2 has a larger raw data size than cube 1. But due to its small geometry, the server chose to send the geometry data instead of raw data.
Case 3: The raw data size in this case is further increased but with similar sizes in geometry and FB to Cases 1 and 2. Sending the geometry data is preferable for interactive visualization because the regeneration of FB introduces additional traffic when the client changes the view parameters.
Case 4: A CT scanned hand data has a raw data size of 81.9 Mbytes. Since volume rendering is employed, we only need to decide whether to send raw data or FB to the client in this case.

5. Conclusion and Future plan

We proposed a framework and an analytical model for mapping visualization pipelines onto computer networks. The dynamic programming method is employed for computing an optimal decomposition and mapping of the visualization pipeline. It would be of future interest to study various other formulations of this problem from the viewpoint of different computational criteria and practical implementation. In future, we plan to include intermediate hosts in our implementation using the dynamic programming algorithms, and also deploy our system over dedicated networks. We also plan to incorporate newer transport methods in our visualization system at a later stage.

Acknowledgements

This research is sponsored by the High Performance Networking Program of the Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC, the Defense Advanced Projects Research Agency under MIPR No.~K153, and by National Science Foundation under Grants No. ANI-0229969 and No. ANI-335185.

References

[1] A. Kaufman, “Trends in visualization and volume graphics”, *Scientific Visualization Advances and Challenges*, IEEE Computer Society Press, 1994.

[2] N. S.V. Rao, Y.C. Bang, S. Radhakrishnan, Q. Wu, S.S. Iyengar, and H. Cho, “NetLets: Measurement-based routing daemons for low end-to-end delays over networks”, in *Computer Communications*, 26, no. 8, pp. 834-844, 2003.

[3] I. Bowman, J. Shalf, and K. Ma, “Performance modeling for grid-based visualization”, submitted to *Parallel Graphics and Visualization* 2004.

[4] E.J. Luke and C.D. Hansen, “Semotus Visum: a flexible remote visualization framework”, *Proc. Visualization 2002*, pp.61-68, 2002.

[5] P. Fränti, O. Virmajoki and T. Kaukoranta, "Branch-and-bound technique for solving optimal clustering", in *Int. Conf. on Pattern Recognition* (ICPR'02), pp.232-235, 2002.

[6] M. Zhu, Q. Wu, N.S.V. Rao, S. S. Iyengar, On optimal mapping of visualization pipeline onto linear arrangement of network nodes, submitted to *Conference on Visualization and Data Analysis*, 2005.