A Novel Parallel Traffic Control Mechanism for Cloud Computing

Presented by Wei Dai
Importance of Traffic Control in Cloud

• Cloud must guarantee all resources as a service and provide them to users according to customized SLAs.

• There may be tens of thousands or even more users accessing resources simultaneously, which results in tremendous network traffic.

• Therefore network bandwidth must be effectively managed to serve all users.
Traffic Control Algorithms

• Some don’t have the concept of “class”, and can not process multiple data flows. (e.g. SFQ and TBF)

• PRIO is an optimization to simple FIFO queue discipline.

• CBQ (Class-Based Queuing) is a network router queuing method that groups traffic into classes and allows traffic to share bandwidth equally.
HTB

• HTB (Hierarchical Token Bucket) is based on CBQ, but having the improvement that unused bandwidth from one class can be used by other classes.

• A mechanism that provides network QoS capabilities

• A network traffic scheduler and shaper used for fine-tuning traffic flow
CBQ vs. HTB

• Both of them use a tree of class nodes, which describes how the network bandwidth is allocated.

• Root node represents the total bandwidth, leaf nodes represent different users, other nodes represent groups of users.

• Every class in CBQ is allocated an Assured Rate (AR), at which it can send data at most.
CBQ vs. HTB

• The advantage of HTB is that it allows classes to borrow bandwidth from parent class.

• Besides AR, every class in HTB also has a Ceil Rate (CR), which is higher than AR.

• Classes in HTB can borrow bandwidth from their parents after they reach AR, but the borrowing cannot break their CR.
CBQ vs. HTB

CBQ (AR)

1Gbps

400Mbps

600Mbps

200Mbps 200Mbps 200Mbps 400Mbps

HTB (AR/CR)

1Gbps

400/600Mbps

600/800Mbps

200/400 Mbps 200/400 Mbps 200/600 Mbps 400/800 Mbps

Figure 1. Comparison between CBQ and HTB.
Limitation of HTB

• The current implementation of HTB has a limit of 0.5Gbps on its capacity, which makes it only suitable for small scale scenarios.

• Two reasons for the limit:
  ▪ HTB has two main operations: enqueue and dequeue. The current HTB is sequential, i.e. every time there can be only one packet to enqueue or dequeue.
  ▪ In the current sequential HTB, there is a chance of concurrent operation on HTB tree, so locks are required, which costs time.
Revised HTB

• Three key features:
  ▪ Parallelization
  ▪ Elimination of lock
  ▪ Lock-free FIFOs

• Parallelization
  Mapping operations of enqueue and dequeue onto different CPU cores and executing them in pipeline
Figure 2. 2-core pipelining using application-level FIFOs
Revised HTB

• Elimination of lock
  Based on the important observation that activating and deactivating the service for a class according to its queue length in the original HTB are not necessary.
Elimination of Lock

Figure 3. *htb_enqueue* using *htb_activate* and *htb_dequeue* using *htb_deactivate*
Elimination of Lock

Figure 4. \( htb\_enqueue \) and \( htb\_dequeue \) with no \( htb\_activate \) and \( htb\_deactivate \)
Revised HTB

• Lock-free FIFOs
  ▪ The head and tail of a FIFO are owned by two different cores and are modified independently. Two arrays are used to separate the head and tail of a FIFO into separate cache lines to avoid cache crash.

  ▪ The FIFO status is detected based on element value instead of the comparison of head value with tail value. NULL indicates an empty queue, and non-NULL indicates a full queue.
Lock-free FIFOs

```c
1:   FIFO_put(FIFO_ELEMENT *data, int i) {
2:     head = queue_head[i];
3:     if (NULL != queue[i][head])
4:         return FALSE;     //! The queue is full
5:     queue[i][head] = *data;
6:     queue_head[i]++;     //! mod add
7:     return TRUE;
8: }

9:   FIFO_get(FIFO_ELEMENT *data, int i) {
10:    tail  = queue_tail[i];
11:    temp  = *data = queue[i][tail];
12:    if (NULL == temp)
13:        return FALSE;    //! The queue is empty
14:    queue[i][tail] = NULL; //! Mark as useable
15:    queue_tail[i]++;      //! mod add
16:    return TRUE;
17: }
```

Figure 5. An array of lock-free FIFOs
Evaluation Design

Figure 6. HTB bandwidth tree in evaluation
Experiments

• Experiment 1 (File-1):
To test whether the revised HTB can work well under extremely high traffic pressure. The sending rates of all leaf classes are beyond their assigned rates.

<table>
<thead>
<tr>
<th>FILE</th>
<th>#Packets</th>
<th>#Pkt Len.</th>
<th>#Max Len.</th>
<th>#Min Len.</th>
<th>#Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>File-1</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-2</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>533</td>
</tr>
<tr>
<td>File-3</td>
<td>9,765,925</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-4</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>File-5</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1067</td>
</tr>
</tbody>
</table>
Experiments

**TABLE I. TRACE files used for HTB**

<table>
<thead>
<tr>
<th>FILE</th>
<th>#Packets</th>
<th>#Pkt Len.</th>
<th>#Max Len.</th>
<th>#Min Len.</th>
<th>#Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>File-1</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-2</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>533</td>
</tr>
<tr>
<td>File-3</td>
<td>9,765,925</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-4</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>File-5</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1067</td>
</tr>
</tbody>
</table>

• Experiment 2 (File-2):
  To test whether the revised HTB can work well when there are many empty packet queues.
Experiments

• Experiments 3 and 4 (File-3):
To test whether the revised HTB can work well in the worst case scenario, where the length of all packets is 64 bytes. The revised HTB is tested in experiment 3, and the original HTB is tested in experiment 4 for comparison.

<table>
<thead>
<tr>
<th>FILE</th>
<th>#Packets</th>
<th>#Pkt Len.</th>
<th>#Max Len.</th>
<th>#Min Len.</th>
<th>#Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>File-1</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-2</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>533</td>
</tr>
<tr>
<td>File-3</td>
<td>9,765,925</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-4</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>File-5</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1067</td>
</tr>
</tbody>
</table>
Experiments

### TABLE I. Trace files used for HTB

<table>
<thead>
<tr>
<th>FILE</th>
<th>#Packets</th>
<th>#Pkt Len.</th>
<th>#Max Len.</th>
<th>#Min Len.</th>
<th>#Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>File-1</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-2</td>
<td>2,397,696</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>533</td>
</tr>
<tr>
<td>File-3</td>
<td>9,765,925</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>File-4</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>File-5</td>
<td>4,795,392</td>
<td>782</td>
<td>1500</td>
<td>64</td>
<td>1067</td>
</tr>
</tbody>
</table>

- Experiments 5 and 6 (File-4 and File-5 respectively):
  To test the scalability of the revised HTB. Similar to Experiment 1 and 2 respectively, but the total bandwidth, the number of total users, and the number of total leaf classes are doubled.
Performance Analysis

### TABLE II. PERFORMANCE OF HTB

<table>
<thead>
<tr>
<th>Exp.</th>
<th>#Trace</th>
<th>#MPPS</th>
<th>#MBPS</th>
<th>#Enq.</th>
<th>#Deq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>File-1</td>
<td>1.29</td>
<td>1008</td>
<td>0.39</td>
<td>0.54</td>
</tr>
<tr>
<td>2</td>
<td>File-2</td>
<td>1.29</td>
<td>1006</td>
<td>0.39</td>
<td>0.57</td>
</tr>
<tr>
<td>3</td>
<td>File-3</td>
<td>14.1</td>
<td>941</td>
<td>0.39</td>
<td>0.53</td>
</tr>
<tr>
<td>4</td>
<td>File-3</td>
<td>6.7</td>
<td>427</td>
<td>0.64</td>
<td>1.11</td>
</tr>
<tr>
<td>5</td>
<td>File-4</td>
<td>2.60</td>
<td>2033</td>
<td>0.39</td>
<td>0.54</td>
</tr>
<tr>
<td>6</td>
<td>File-5</td>
<td>2.59</td>
<td>2026</td>
<td>0.39</td>
<td>0.58</td>
</tr>
</tbody>
</table>
Figure 7. Output traffic rate of total traffic
Performance Analysis

Figure 8. Output traffic rate of a selected user
References

• A Novel Parallel Traffic Control Mechanism for Cloud Computing
  Zheng Li, Nenghai Yu, Zhuo Hao

• HTB Homepage
  http://luxik.cdi.cz/~devik/qos/htb/