Load Balancing in Dynamic Structured P2P System

B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica

Ankit Pat

November 19, 2013

Outline

- Introduction
- Preliminaries
- Background
- Load Balancing Algorithm
- Empirical Evaluation
- Conclusion
- Discussion

Introduction

- Most DHT supporting P2P systems distribute objects (data) randomly among nodes
- Some nodes have $\Theta(log\ N)$ imbalance
- Other factors resulting in imbalance
  - non-uniform distribution of objects in ID space
  - heterogeneity in object loads
  - node capacities
  - variability of a node's load with time

Introduction

- This paper proposes the first algorithm for dynamic load balancing in heterogenous, structured P2P systems
    - data items inserted/deleted continuously
    - nodes join/depart continuously
- Conducts extensive simulations to show its validity

Definitions

- **Load:** Represents the needed storage space, popularity, needed processor time etc. of the object
- **Movement Cost:** Cost associated with moving an object between nodes
- **Capacity:** Each node has a fixed capacity for e.g. disk space, processor speed, bandwidth etc.
- **Node Utilization:** Total load divided by capacity for a node
- **System Utilization:** Total load across nodes divided by total capacity of all nodes

Goals

- Minimizing the load imbalance across nodes
- Minimizing the amount of load moved between nodes

Virtual Servers

- Most DHTs map a region of the ID space to a node
- Unique IDs are attached to the object and the responsible node in the same ID space
- With virtual servers, this mapping is done on virtual servers instead of node
- A node now has multiple virtual servers and hence IDs
- No need to change underlying DHT with joining/departing of nodes (advantage)

Static Load Balancing Techniques

- *One-to-one scheme*: lightly loaded node periodically contacts a node at random
- *One-to-many scheme*: a heavy node contacts a directory node which is contacted by random light nodes
- *Many-to-many scheme*: each directory maintains load information of a set of heavy & light nodes

Node

`Node`(time period T, threshold k_e)

- *Initialization:* Send $(c_n, \{\ell_{v_1}, \ldots, \ell_{v_m}\})$ to `RandomDirectory()`
- *Emergency action:* When u_n jumps above k_e:
  1) Repeat up to twice while $u_n > k_e$:
  2) $d \leftarrow$ `RandomDirectory()`
  3) Send $(c_n, \{\ell_{v_1}, \ldots, \ell_{v_m}\})$ to d
  4) `PerformTransfer`(v, n') for each transfer $v \rightarrow n'$ scheduled by d
- *Periodic action:* Upon receipt of list of transfers from a directory:
  1) `PerformTransfer`(v, n') for each transfer $v \rightarrow n'$
  2) Report $(c_n, \{\ell_{v_1}, \ldots, \ell_{v_m}\})$ to `RandomDirectory()`

Directory

`Directory`(time period T, thresholds k_e, k_p)

- *Initialization:* $I \leftarrow \{\}$
- *Information receipt and emergency balancing:* Upon receipt of $J = (c_n, \{\ell_{v_1}, \ldots, \ell_{v_m}\})$ from node n:
  1) $I \leftarrow I \cup J$
  2) If $u_n > k_e$:
  3) $\quad reassignment \leftarrow \texttt{ReassignVS}(I, k_e)$
  4) $\quad$ Schedule transfers according to $reassignment$
- *Periodic balancing:* Every T seconds:
  1) $reassignment \leftarrow \texttt{ReassignVS}(I, k_p)$
  2) Schedule transfers according to $reassignment$
  3) $I \leftarrow \{\}$

Reassignment of Virtual Servers

`ReassignVS`(Load & capacity information I, threshold k)

1) *pool* $\leftarrow \{\}$
2) For each node $n \in I$, while $\ell_n / c_n > k$, remove the least loaded virtual server on n and move it to *pool*.
3) For each virtual server $v \in$ *pool*, from heaviest to lightest, assign v to the node n which minimizes $(\ell_n + \ell_v)/c_n$.
4) Return the virtual server reassignment.

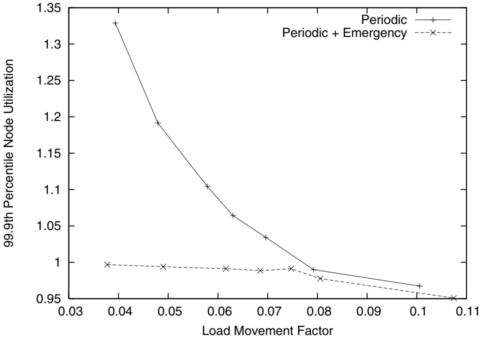
Some Design Issues

- *Periodic vs. emergency balancing*: large T is preferred but emergency situations are taken care of
- *Choice of parameters*: threshold k_e is set to 1 and k_p is set to $(1+\hat{\mu})/2$
- *Stale information*: 'node reporting times' across directories is not synchronized

Metrics

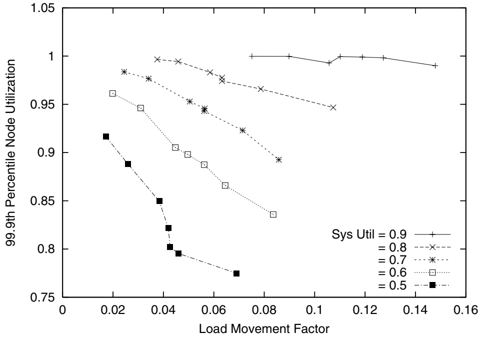
- *Load Movement Factor*: total movement cost due to load balancing divided by the total most of moving all objects in the system once
- *99.9th percentile node utilization*: maximum over all simulated times t of the 99.9th percentile of node utilizations at time t

Basic Effect of Load Balancing



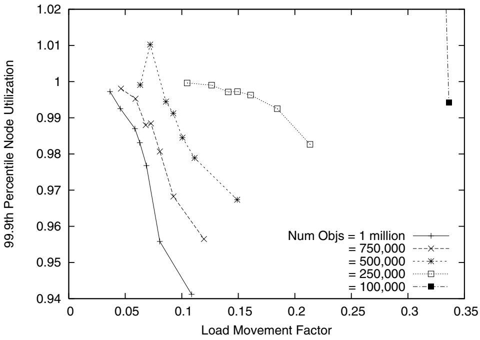
- Tradeoff between load movement and 99.9th percentile node utilization
- Rest of the simulations have emergency balancing is enabled

Load Movement vs. 99.9th Percentile Node Utilization



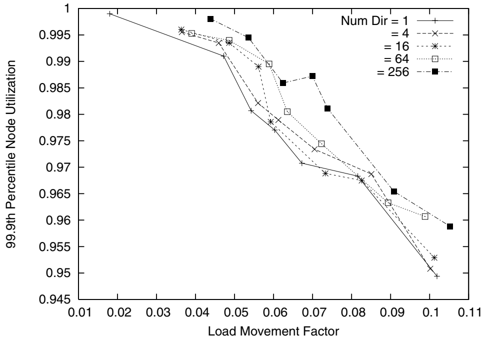
- 99.9% of nodes are underloaded for load movement factor <0.08

Load Movement vs. 99.9th Percentile Node Utilization



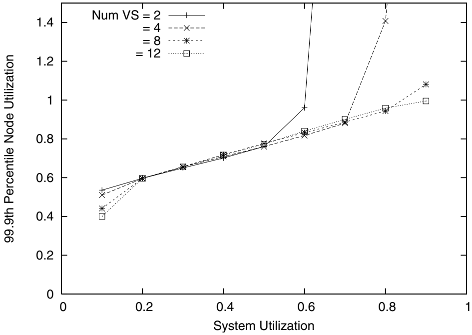
- For at least 250,000 objects, good load balance and load movement factor of 0.11 is achieved

Load Movement vs. 99.9th Percentile Node Utilization



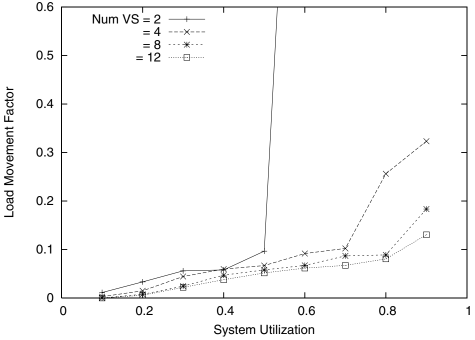
- Number of directories has a small impact on the metrics

Number of Virtual Servers



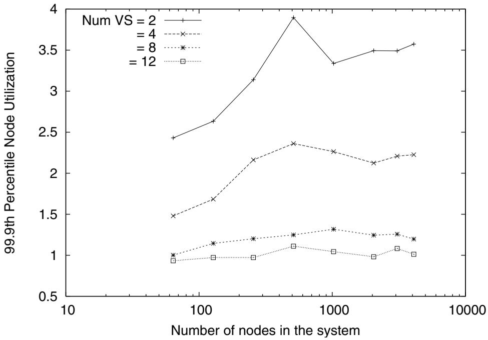
- 99.9th percentile node utilization increases roughly linearly with system utilization

Number of Virtual Servers



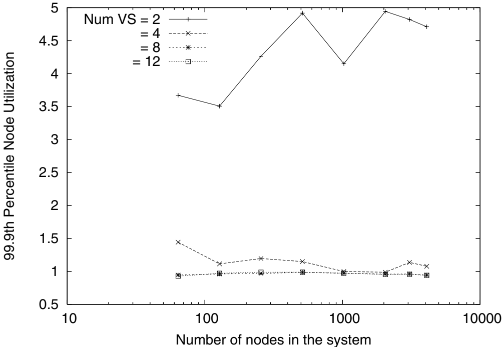
- Increase in virtual servers looks good for load movement factor

Heterogenous Node Capacities



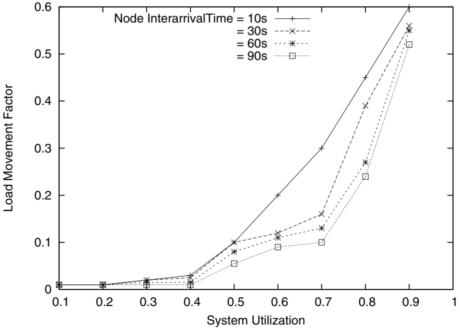
- Uses homogeneous node capacities and number of virtual servers
- Grows in 99.9th percentile of nodes roughly linear in log N

Heterogenous Node Capacities



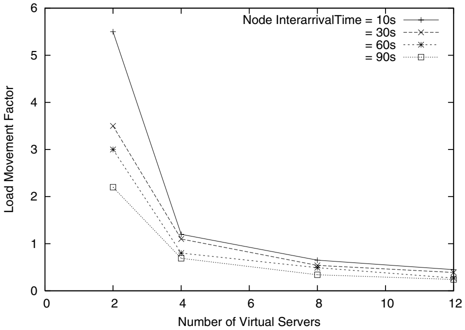
- Uses heterogeneous capacity distribution
- Achieves remarkable decrease in 99.9th percentile node utilization with growth in *N*

Node Arrivals and Departures



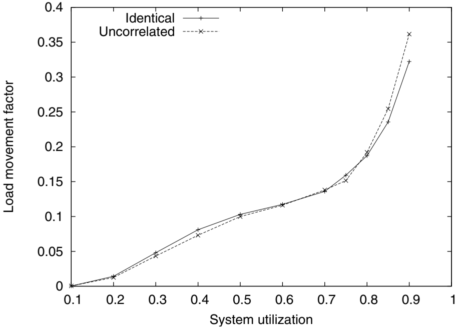
- Load moved by the load balancer as a fraction of the load moved by DHT vs. system utilization
- For the default 12 virtual servers per node, the algorithm never moves more than 60% of the load compared to DHT.

Node Arrivals and Departures



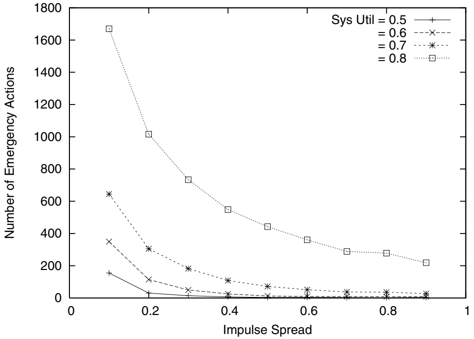
- Load moved by the load balancer as a fraction of the load moved by the DHT vs. number of virtual servers

Object Movement Cost



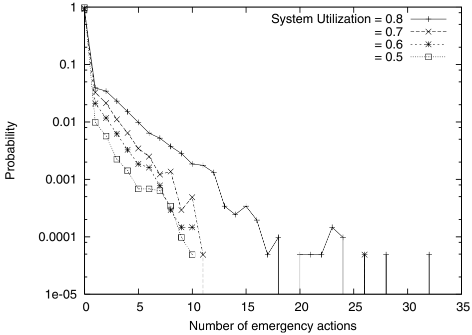
- Load movement factor vs. system utilization for two cases of object load and object movement cost

Non-uniform Object Arrival Patterns



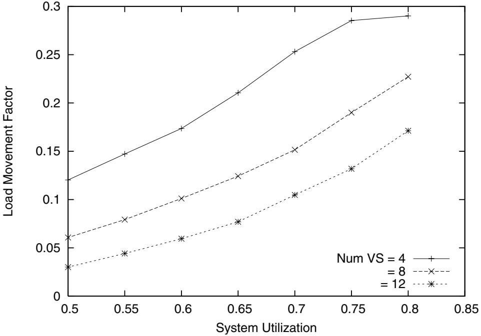
- “Impulse” refers to objects in a contiguous interval in the ID space with aggregate load equalling 10% of total system load
- Objects arrival is tuned so that periodic load balancing does not run while emergency load balancing may be invoked

Non-uniform Object Arrival Patterns



- PDF of number of emergency actions taken after an impulse of 10% concentrated in 10% of the ID space

Non-uniform Object Arrival Patterns



- Load movement factor vs. system utilization after an impulse in 10% of the ID space

Conclusion

- Proposed a load balancing algorithm for dynamic, heterogeneous P2P systems
- Heterogeneity implies varying –
  - object loads
  - node capacity
  - continuous insertion and deletion of objects
  - skewed object arrival patterns
  - continuous arrival/departure of nodes
- Achieves load balancing for system utilizations of 90% while moving only 8% of the arriving load
- Moves less than 60% of the load the underlying DHT moves for node arrivals and departures
- Heterogeneity can help improving scalability

Discussion

1. Why are the times at which the nodes report to the directories not synchronized?
2. Glich in technical presentation in the “Load Balancing Algorithm” section!
3. How about reporting Directory utilization in Node(T, k_e)
4. Possible usage of Kalman Filters?

THANK YOU!