Rebalancing vBucket Mapping

2: The Topology-Replication-Mapping Scheme

Weikang Zhou

Couchbase

July 27, 2012

Table of contents

Notations

- N: number of vbuckets
- M: number of nodes
- L: number of copies for each vBucket
- S: slave number for each node

Mapping A

vBucket Mapping matrix A: $N \times L$ matrix of node IDs.

Count R

Replication matrix R: $M \times M$ matrix of occurrences of replication pairs.

Topology RI

Adjacency matrix RI: $M \times M$ matrix. $RI := (R > 0)$

Previous Algorithm Recap

The Replication-Mapping scheme:

$$\boxed{\begin{matrix} R \\ \text{Replication} \end{matrix}} \Longleftrightarrow \boxed{\begin{matrix} A \\ \text{Mapping} \end{matrix}}$$

- Input: balanced mapping with known node ordering
- Node ordering determines the (known) optimal topology of target mapping, which determines R
- The algorithm then tries to change A minimally so that R will conform to target R
- Weakness
  1. Must take a balanced input mapping
  2. Must keep track of the original ordering of nodes
  3. Can only generate one type of topology
  4. Cannot extend to accommodate other constraints

Algorithm: new scheme

The Topology-Replication-Mapping scheme:

RI Topology	$\Longleftrightarrow$	R Replication	$\Longleftrightarrow$	A Mapping

- Each block specializes in optimizing different aspects:
  1. RI (topology): Slave nodes selection, node tags, node health constraint, (relaxed slave number constraint), ...
  2. R (replication count): balancedness, (node heterogeneity), ...
  3. A (vBucket mapping): movement minimization
- New algorithm: current $A \longrightarrow$ current $R \longrightarrow$ current $RI \longrightarrow$ target $RI \longrightarrow$ target $R \longrightarrow$ new A

Finding optimal topology

- Current $RI \longrightarrow$ target RI: the most computationally expensive step.
- Start from any feasible topology and greedily look for the optimal topology.
- Proposal: uniformly randomly sample two rows and two columns having two 1's on the diagonal, and propose to switch $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- Evaluation the "energy" of a topology: for a particular RI, we compute its energy to approximate the distance between the resulting target R with the original R.
- We accept the proposal if and only if energy decreases.
- $\approx 10^6$ proposals are assessed in a typical setting: energy evaluation should be effective and *fast*.

Energy evaluation

- Energy $E := \sum_{i,j} \left| R(i,j) - \widetilde{R}^T(i,j) \right|$, where $\widetilde{R}^T := \frac{N \cdot L}{M \cdot S} RI$
- Consider the following difference with the same energy:
  1. $\begin{bmatrix} & n & \\ n & -n & \\ & & -n \end{bmatrix}$ (requiring $3n$ steps to balance)
  2. $\begin{bmatrix} & n & \\ n & & -n \\ & -n & \end{bmatrix}$ (requiring only $2n$ steps to balance)
- Our algorithm, however, is indifferent between the two.
- This can arise when we are simultaneously adding and deleting nodes.
- For now, to circumvent this issue, we "cheat" by matching the deleted nodes with added nodes first.
- A better energy function??

Target $RI \longrightarrow$ target R

- Nonzero positions are determined by target RI
- We *disperse* the nonzero elements in target R by fixing row sums first, so that target R will be as close to the previous R as possible.
- Then we only have to balance the column sums of R, as this is the only remaining constraint to be satisfied.
- We search to switch two elements in a row, or switch two element in different rows via a third column, or weakly switch between elements in a row.
- If needed, we revisit row sums and search to switch two row sums.
- Here RI need not be "connected" any more, and this can be problematic...

Target $RI \longrightarrow$ target R

For a given (optimal) topology RI, it is *not* always possible to find R such that its column sums are balanced. ($\approx 0.3\%$ cases for $L = 4$.)

R	0	128	129	128	128	128	128	128	128	128	129	128	127	128	128	128	128	127	128	128	128	128	128	128	128
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	0	13	12	13	13	13	13	13	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
126	0	13	0	13	13	13	12	13	12	12	12	13	0	0	0	0	0	0	0	0	0	0	0	0	0
126	0	13	13	0	13	13	12	12	13	12	13	12	0	0	0	0	0	0	0	0	0	0	0	0	0
126	0	13	12	13	0	13	13	12	12	13	13	12	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	13	13	13	12	0	13	13	13	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	13	13	13	13	12	0	13	13	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	12	13	13	13	13	13	0	13	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	13	13	12	13	13	13	13	0	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	13	13	13	13	12	13	13	13	0	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	13	13	13	12	13	13	13	13	13	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	12	13	13	13	13	13	13	13	13	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
129	0	0	0	0	0	0	0	0	0	0	0	0	0	13	13	0	0	12	13	13	13	13	13	13	13
126	0	0	0	0	0	0	0	0	0	0	0	0	12	0	13	13	13	0	0	13	12	12	13	12	13
126	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	13	13	12	12	12	12	13	13	13	0
129	0	0	0	0	0	0	0	0	0	0	0	0	13	12	13	0	13	13	0	13	0	13	13	13	13
126	0	0	0	0	0	0	0	0	0	0	0	0	12	13	12	12	0	12	13	13	0	13	13	0	13
129	0	0	0	0	0	0	0	0	0	0	0	0	13	13	13	0	13	0	12	13	13	13	13	13	0
126	0	0	0	0	0	0	0	0	0	0	0	0	0	12	13	13	13	13	0	13	13	12	12	0	12
126	0	0	0	0	0	0	0	0	0	0	0	0	13	13	0	13	12	13	13	0	13	0	12	12	12
129	0	0	0	0	0	0	0	0	0	0	0	0	13	13	12	13	13	13	13	0	0	13	0	13	13
129	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	13	12	13	13	13	13	0	13	13	13
129	0	0	0	0	0	0	0	0	0	0	0	0	13	13	13	13	13	0	13	12	13	0	0	13	13
129	0	0	0	0	0	0	0	0	0	0	0	0	12	0	13	13	13	13	13	13	13	13	0	0	13
129	0	0	0	0	0	0	0	0	0	0	0	0	13	13	13	12	0	13	13	0	13	13	13	13	0

Target $R \longrightarrow A$

- Two step greedy search:
  1. **Inter-row change**: change of one active node modifies two row sums of R by $(L-1)$.
  2. **Intra-row change**: change of one replica node modifies two column sums of R by 1.
- We minimize the movements of active node first; then we minimize the movements of replica nodes.
- We assume there is no trade-off between the two: the above two steps are separate.
- The "no trade-off" assumption is only valid for small L.
- Currently the implementation is for indexed nodes, i.e. promoting a replica to active is counted as one move.

Lower bound

- A theoretical lower bound evaluates the performance of the algorithm
- The lower bound should be independent of the topology chosen and should apply to any input mapping
- Recall that:
  1. Balanced target R^T has row sums and column sums $\approx \dfrac{N(L-1)}{M}$.
  2. Change of an active node in A changes two row sums of R by $(L-1)$.
  3. Change of a replica node in A changes two column sums of R by 1.
- Lower bound =
$$\min\left\{\frac{1}{2(L-1)}\sum_i \left|R \text{ row sum of row } i - R^T \text{ row sum of row } i\right| + \frac{1}{2}\sum_i \left|R \text{ column sum of row } i - R^T \text{ column sum of row } i\right|\right\}$$
- This is more conservative than the previous $\dfrac{N \cdot L}{\max\{M, M_0\}}\Delta M$.

Diagnostic

1293.6	1266	461	12	585
1293.6	1266	461	15	589
1293.6	1266	461	14	589
1293.6	1266	460	8	566
1293.6	1266	459	4	563
1293.6	1266	460	13	577
1293.6	1266	460	16	591
1293.6	1266	456	20	**594**
1293.6	1266	458	15	581
1293.6	1266	460	14	578
1293.6	1266	459	10	576
1293.6	1268	456	9	581
1293.6	1266	456	11	575
1293.6	1266	461	11	574
1293.6	1266	458	9	576
1293.6	1266	456	5	573
1293.6	1268	461	9	592
1293.6	1266	456	14	569
1293.6	1266	461	19	581
1293.6	1266	461	10	573
1293.6	1266	461	10	559
1293.6	1266	461	8	571
1293.6	1266	460	8	569
1293.6	1266	460	9	**559**
1293.6	1266	457	11	562
1293.6	1266	459	12	573
1293.6	1266	456	11	573
1293.6	1266	461	14	577
1293.6	1268	458	9	574
1293.6	1266	459	3	559
1293.6	1266	458	11	562
1293.6	1266	457	13	566

- Input mapping: random permutations of vBuckets with the same mapping
- $L = 4, M : 23 \to 26$, lower bound 452.
- 1st column: final minimum energy level
- 2nd column: after balancing target R^T, $\sum |R(i,j) - R^T(i,j)|$
- 3rd column: lower bound computed through R^T
- 4th column: additional move in $R \to A$
- 5th column: actual number of movements
- Ideally, all columns should be the same

Testing results: fixed ΔM

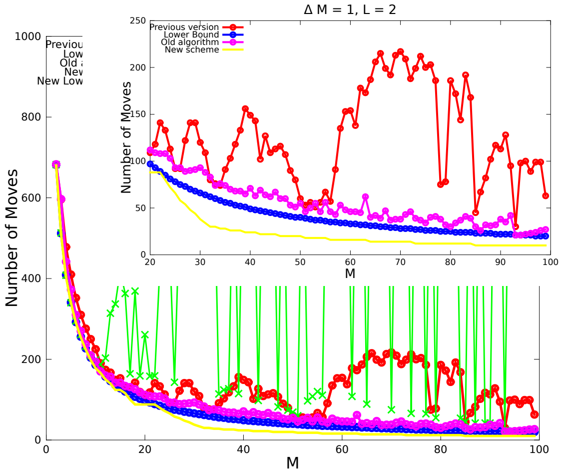
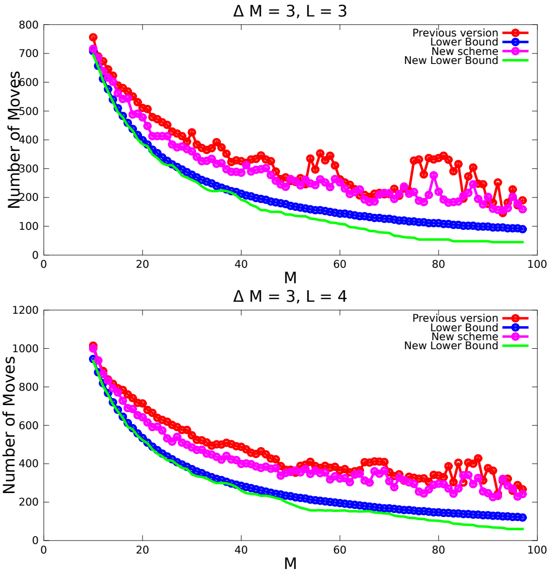


Figure: $\Delta M = 1, L = 2$

Testing results: fixed ΔM



Testing results: fixed M

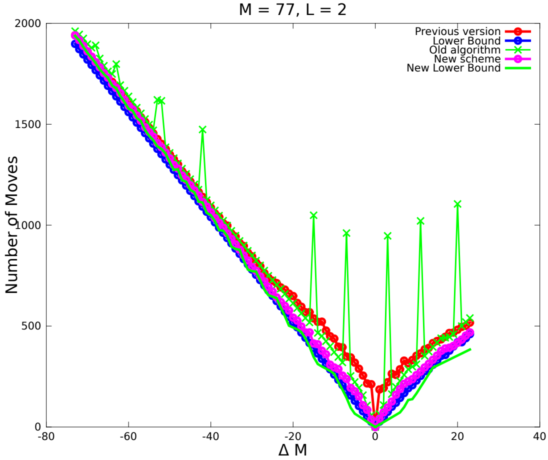


Figure: $M = 77, L = 2$

Testing results: fixed M

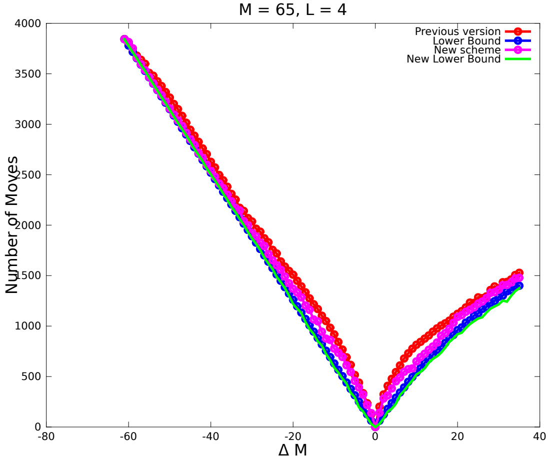
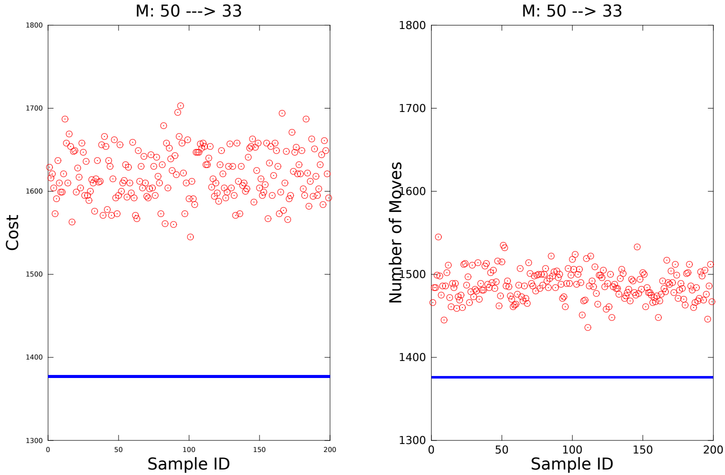


Figure: $M = 65, L = 4$

Testing results: variance in deletion



(a) Previous algorithm (b) New algorithm

Testing results: choosing S

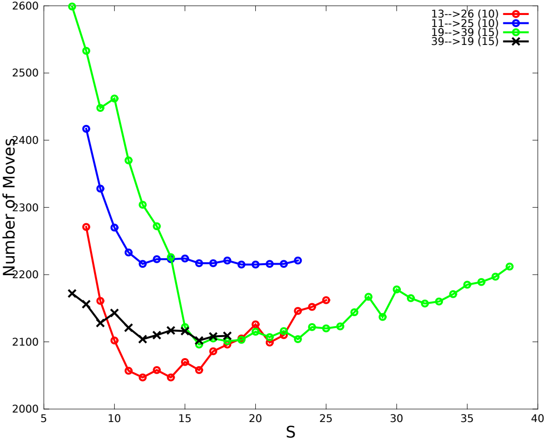
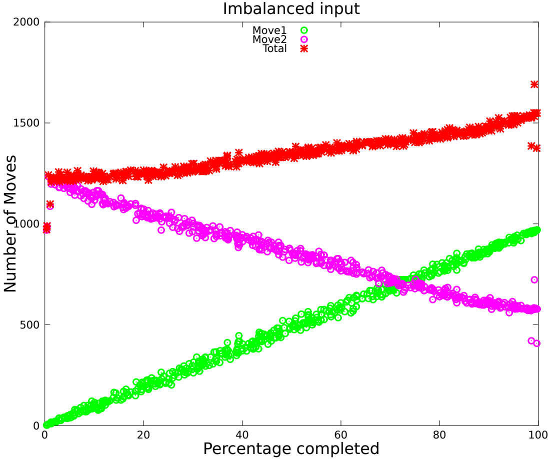


Figure: Number of movements versus different sizes of S

Testing results: imbalanced input



- Stage 1: 21 nodes, add 5 and delete 3
- Stage 1 is incomplete
- Stage 2: add 2 more nodes, delete 3 (2 of them deleted in stage 1)

Next step

1. Non-indexed version.
2. Nodes with tags (eg. shelves)
3. Unhealthy nodes
4. Heterogeneous nodes.