A Constraint Programming Algorithm for Assigning Replicas to Applications

Tyrel Russell

University of Waterloo

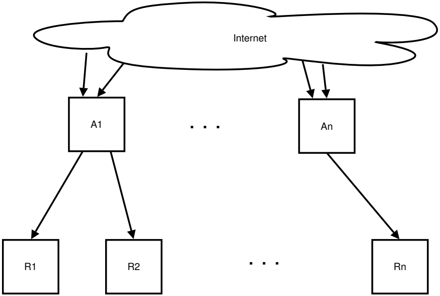
Outline

Clustered web applications

- Peak load often exceeds the capacity of a single server
- But provisioning for peak load is inefficient
- Solution: Dynamically assign resources from a pool based on the changing load of the system
- Problem: Finding an optimal assignment is difficult even in a static environment

The Application Model

- n applications $A_1, A_2, \ldots, A_n$
- m replicas (or machines) $R_1, R_2, \ldots, R_n$
- Associated with each application A_i is a load L_i and a latency requirement l_i
- Associated with each replica R_j is capacity q_j



Model Assumptions

- The replicas are of fixed size
- Each replica is only assigned to a single application
- Every query can be executed in a fixed time by a given replica and that time is $\frac{1}{q_j}$
- There are a sufficient number of replicas to handle all of the load in the system
- We assume that there will be a number of replicas (possibly all) that are of the same quality

Outline

Assigning Replicas to Applications

- Each application should meet its latency requirement l_i
- Therefore, every query must be satisfied in under l_i seconds
- Over a given interval t, we need to assure that no query exceeds the latency requirement
- This requires us to find the peak load of an interval

Finding a Partition

- We assume there is some partitioning of replicas P
- For each application A_i, we have $P_i = \{j | j \in R \wedge R_j = A_i\}$
- And $\bigcup_{i \quad A} P_i = R$

Cost of a Partition and the Constraint

- Once we have a partition, we need to define the cost of the partition
- Let $w_j = q_j l_i$ be the number of queries that can be executed in l_i seconds
- The load that can be handled by the system is $\sum_{j \ \ P_i} w_j$
- This is equal to

$$\sum_{j \ \ P_i} w_j = \sum_{j \ \ P_i} l_i q_j = l_i \sum_{j \ \ P_i} q_j = l_i Q_i$$

- To have a feasible solution, we need $L_i^p \leq l_i Q_i \Rightarrow \frac{L_i^p}{Q_i} \leq l_i$

Outline

1. Introduction
2. The Replication Assignment Problem
3. The Optimization Problem
4. The Homogeneous Solution
5. The Heterogeneous Solution
6. Conclusion

The Objective Function and the Optimiztion

- From the constraint, we see that we want to make sure $\frac{L_i^p}{Q_i}$ does not approach l_i
- Therefore we wish to minimize,

$$\sum_{i \quad A} \frac{L_i^p}{Q_i}$$

- such that $\frac{L_i^p}{Q_i} \leq l_i, \forall_{i \quad A}$.

Outline

The Homogeneous Model

- Note if all replicas are identical, $q_j = q_k = q$ for all j and k
- Therefore, for a given application A_i and P_i, we have

$$Q_i = \sum_{j \quad P_i} q_j = \sum_{j \quad P_i} q = \|P_i\| q$$

- For convenience, we will let $\|P_i\| = r_i$
- Therefore, we have $Q_i = r_i q$

Upper and Lower Bounds

- For a lower bound, we note that $Q_i \geq \frac{L_i^p}{l_i}$
- Using the homogeneous assumption, we have

$$r_i q \geq \frac{L_i^p}{l_i} \Rightarrow r_i \geq \frac{L_i^p}{q l_i}$$

- Therefore, the lower bound for r_i is $\frac{L_i^p}{q l_i}$
- The upper bound is

$$ub(r_i) = r_i + (m - \sum_{j \quad A} lb(r_j)) = r_i + (m - r\ )$$

Marginal Gains

- Want to find the the number between the upper and lower bounds that minimizes the objective function.
- Need $lb(r_i)$ replicas to satisfy the constraints for each application A_i
- Note the marginal gain of adding a replica can be calculated as

$$\frac{L_i}{(r_i + (e-1))q} - \frac{L_i}{(r_i + e)q}, e \geq 1$$

- If we calculate these values for all $m - r$ values of e for every application

The Homogeneous Solution

- If we order the marginal gains in descending order, we can determine the $m - r$ largest marginal gains
- By counting the multiplicity of marginal gains for each application A_i, we determine a number e_i for each application
- The size of the partition of each application should therefore contain $r_i = \frac{L_i^p}{ql_i} + e_i$
- This is a tight bound and therefore any solution that meets this condition is a homogeneous solution

Outline

The Heterogeneous Model

- In the heterogeneous model, the number of possible marginal gains is exponential.
- Therefore, we must approach the problem in a different manner.
- In the heterogeneous model, $\exists_{j,k} q_j \neq q_k$ necessarily
- However, there will be many equivalent classes of replicas

Upper and Lower Bounds

- The upper bound of Q_i is defined as,

$$lb(Q_i) = \frac{L_i}{l_i}$$

- The upper bound of Q_i is defined as,

$$ub(Q_i) = \frac{L_i}{l_i} + \left[\sum_{k \quad R} q_k - \sum_{i,j \quad A} \frac{L_j}{l_j} \right]$$

- We can update the upper bound when replicas exceed their lower bound as,

$$uub(x) = \begin{cases} \frac{L_i}{l_i} + \left[\sum_{k \quad R} q_k - \sum_{j=i, i,j \quad A} Q_j \right] & , Q_j > lb(Q_j) \\ \frac{L_i}{l_i} + \left[\sum_{k \quad R} q_k - \sum_{j=i, i,j \quad A} \frac{L_j}{l_j} \right] & , Q_j \leq lb(Q_j) \end{cases}$$

Consistency Checks and Propagators

- To enforce the bounds, we can make consistency checks
- Globally, the total number of replicas remaining unassigned must be equal to the number of replicas needed to satisfy lower bounds

$$\begin{aligned} Q_{needed} \geq Q_{avail} &\Rightarrow \textit{consistent} \\ Q_{needed} < Q_{avail} &\Rightarrow \textit{inconsistent} \end{aligned}$$

- Locally, an application is in an inconsistent state if there are not enough replicas with the application in its domain

The CSP Formulation

$$\sum_{i \quad A} \frac{L_i}{Q_i}$$

is minimized and,

$$\begin{array}{rc}
 & \forall_{i \quad A} \exists_{j \quad R} R_j = A_i \\
\wedge & \forall_{i \quad A} Q_i \leq ub(Q_i) \\
\wedge & \forall_{i \quad A} Q_i \leq uub(Q_i) \\
\wedge & \forall_{i \quad A} \sum_{j \quad R \quad A_i \quad dom(R_j)} q_j \geq \hat{q}_i
\end{array}$$

Incremental Improvements

- In the next interval, we need to solve the same problem but with a new load L_i
- Instead of solving the system from scratch, we want a minimal solution with as few movements as possible
- We will attempt to move replicas between applications to find a minimal solution in the least number of moves
- To aid in this search we will define (and update) lower and upper bounds on the number of search moves

Future Work

- Relax restriction that all queries must be processed in the same amount of time
- Extend work to deal with overloaded allocations through the use of an error parameter
- Perform experimental evaluations to prove feasibility

Outline

- Presents and optimal solution for the replication assignment problem
- Uses a greedy algorithm to solve the homogeneous case
- Uses a CSP to solve the harder heterogeneous problem