Contents

1 Introduction

There are two methods to model a network:

- create a database of port connections
- create autonomous switches which direct packets to other components

For this model I will be using a database of connections.

1.1 problem

All possible compute bottlenecks:

- CPU clock rate
- memory (L1, L2, RAM, disk) access latency
- memory (L1, L2, RAM, disk) access bandwidth
- network bandwidth
- network latency

We want an efficient and reasonable all-to-all network.

An optimal all-to-all design would have each compute node with $N-1$ ports and no switches (0 hops). For $N > 1000$ this simplisitic approach is not feasiable. Thus we introduce switches.

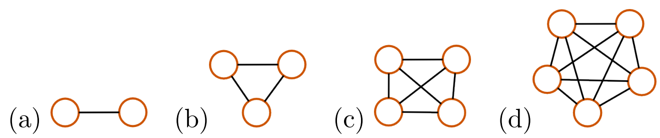


Figure 1: increasing the number of nodes. Doesn't scale well due to limited number of ports on each compute node ($n < N$).

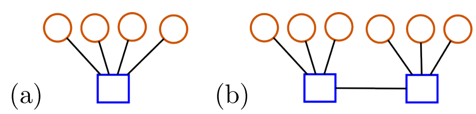


Figure 2: adding switches. Doesn't scale well due to limited number of ports per switch ($m < N$). Also, conjestion (a,b) and bisection bandwidth (a) are poor.

To reduce conjestion and increase bisection bandwidth, add more switches. Also increase the number of ports per compute node. Then the optimization problem becomse non-trivial.

1.2 objectives

If I can show a given topology is better than some other, that is useful.

If I can show how one switch versus another (differentiated by number of ports) affects performance, that is useful.

Note: although this model is useful for all-to-all communication measurement, there are not inherent constraints as far as applying this same model to other interests. You can use whatever fitness function you can define.

1.3 Does Topology matter? A toy model example

Suppose now we have $N = 5$ compute nodes with 1 port each. Then the optimal network design (fewest hops) is to have a 5 port switch:

The other extreme would be to use five of these same switches with only one compute node per switch: Clearly we are spending too much money on switches for the same number of compute nodes. However, this increased hop count (2) also lowers conjestion.

As a compromise, we can use two switches and increase the number of compute nodes:

Notice a few constraints were followed:

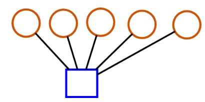


Figure 3: Here number of hops is 1 for each compute node, with 10 pairs (=5*4/2).

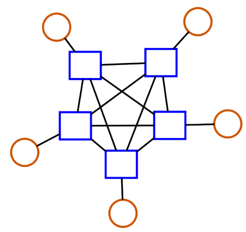


Figure 4: Here number of hops is 2 for each compute node.

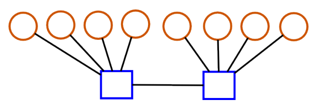


Figure 5: 8 nodes and 2 switches: 12 pairs with 1 hop, 16 pairs with 2 hops.

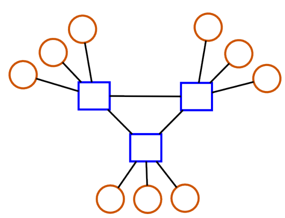


Figure 6: 1 or 2 hops, 9 nodes and 3 switches.

- each switch has same number of ports
- each compute node has one port
- each switch is fully occupied
- each node can reach every other node
- each switch has at least one computer connected to it

The number of permutations increases when we have more than 9 compute nodes and only 5 ports. Even worse, consider when there are multiple ports per computer (but much less than the number of computers).

The parameter space includes

- number of computers
- number of ports per computer
- number of switches
- number of ports per switch

Metrics:

- hop count for each pair
- bisection bandwidth

1.4 Permutations

The number of unique pairs on a network swith N computers is $N(N-1)/2$.

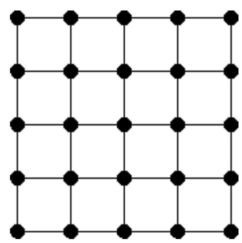
When $N = 4$ then there are 6 pairs. $N = 100$ is 4950 pairs. $N = 10,000$ is 49,995,000 pairs.

2 standard networks

See http://www.cs.nmsu.edu/ pfeiffer/classes/573/notes/topology.html

2.1 Mesh

Mesh (and the related torus) can be of n dimensions, commonly n =2, 3, 6. Useful for physical sciences due to local communication (nearest neighbors). Mesh networks are well-characterized. "Meshes have O(n) cost, O(sqrt(n)) bisection bandwidth, O(n) aggregate bandwidth, and O(sqrt(n)) latency."



2.2 Hypercube

"the latency is is O(log N). There are N processors, each with log2N interfaces, so the cost is O(N log N). and all the processors can use their links simultaneously, so our aggregate bandwidth is O(N). The bisection bandwidth is O(log N)."

2.3 Omega

"scales as $n \log n$."

2.4 Fat Tree

http://en.wikipedia.org/wiki/Fat_tree

2.5 Flattened Butterfly

2.6 Dragonfly

http://research.google.com/pubs/pub34926.html

Fractal

2.7 Clos Network

http://en.wikipedia.org/wiki/Clos_network

2.8 randomly-connected networks

For networks supporting physical models (i.e., mesh), it makes sense to think about dimension, perimeter/surface area, area/volume. This may not apply to scale-free topologies.

If the topology turns out to be scale free, then we wouldn't need to model 1E6 endpoints (that is desirable).

3 more than one port per endpoint

If each endpoint has only one network connection, then we can model a switch-only network. A switch with 4 ports not connected to other switches would have 4 endpoints.

This might simplify the analysis, but here we model compute nodes with more than one port. Then it is necessary to include both "switch" and "compute" type nodes.

4 random network creation

For a given {(number of computers), (number of ports per computer), (number of ports per switch)}, should random computers be plugged into random port switches, or should random switches be connected first?

"connections" database methods:

- each switch is an sub-array of the connections array. The elements of each sub-array denote which computer the switch is plugged into.
- connections pairs: computer–switch and switch-switch. The connections array has sub-arrays of size 2 for each edge of the graph (nodes are either computers or switches).
  - unordered pairs of positive (switch) and negative (computer) integer indices

Features needed:

- supports switches having arbitrary port count (not all switches must have same number of ports

Whether local symmetry (same number of computers plugged into each switch) is a hinderance, benefit, or irrelevant is not clear to me.

Random connections lead to unexpected paths. This could be good, bad, or inconsequential.

5 route enumeration

1. For each computer, see what other computers are available on the same switch (1 hop)
2. For each computer, see what other computers are two switches away (2 hops)
3. ...

When a switch has had all of its computers touched for a given iteration, then we should mark that switch as "touched" (doesn't need to be queried again for current iteration). That is, mark a switch to indicate "all locally-attached computers have number of hops known." This should reduce search time.

6 Maximum and Minimum number switches needed

How many switches (M) are needed when each switch has m ports? (Assume $m < (N-1)$.)

Variables:

- n = number ports per compute node

- N = number of compute nodes
- m = number of ports per switch
- M = number of switches

Observe that we have already made the simplifying assumption of identical compute nodes and identical switches.

6.1 one port per compute node

For N compute nodes, how many switches M are needed when each switch has m ports? We assume there are more compute nodes than ports on one switch ($m < (N-1)$). We must obey the constraints (every compute node must be able to reach every other compute node – "fully connected network"), (all switch ports are used), (all compute node ports are used).

Solution 1: to get an all-to-all network, each node should connect to a tree with $(N-1)$ endpoints (bisection bandwidth is thus maximized). To acheive this, start with one compute node and a switch with m ports (for this example let $m = 5$). At this tree level ($k = 1$) we have $m = 5$, $M' = 1$, and $N - 1 = 4$. (M' refers to the number of switches for this one compute node).

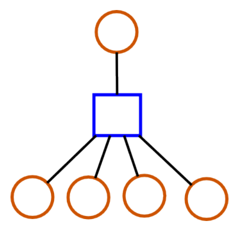


Figure 7: $m = 5$, $M' = 1$, $N - 1 = 4$ (tree level $k = 1$).

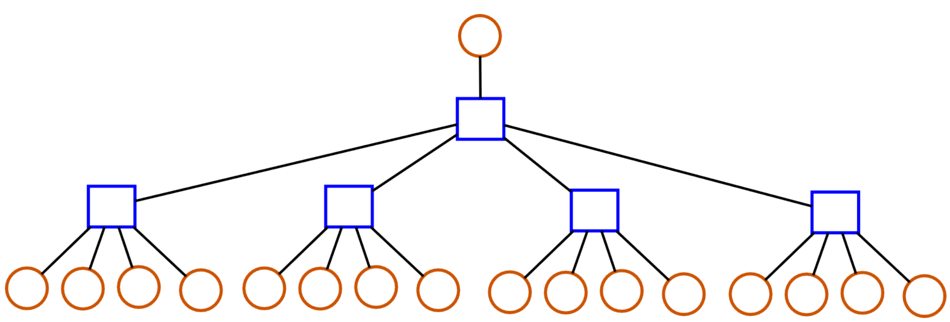


Figure 8: $m = 5$, $M' = 1 + 4$, $N - 1 = 4 * 4$ (tree level $k = 2$).

Generalizing, the number of compute nodes the one we are dealing with can connect to is

$$N - 1 = (m-1)^k \tag{1}$$

and the number switches this one compute node needs to create $N - 1$ endpoints is

$$M' = \sum_{a=1}^{k} (m-1)^{(a-1)} \tag{2}$$

Since there are a total of N compute nodes, the total number of switches M needed is

$$\boxed{M_{max} = N \sum_{a=1}^{k} (m-1)^{(a-1)}} \tag{3}$$

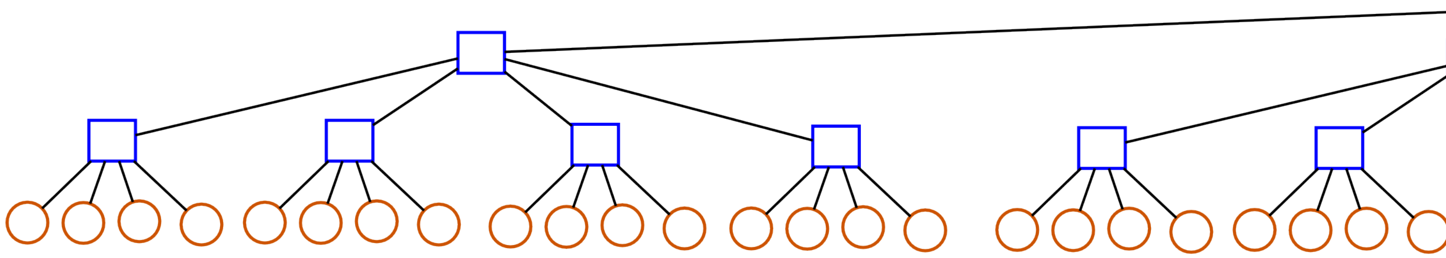


Figure 9: $m = 5$, $M' = 1 + 4 + 4^2$, $N - 1 = 4^3$ (tree level $k = 3$).

This is the maximum number of switches needed.

We can solve for k from Eq. 1

$$\log(N-1) = k \log(m-1) \tag{4}$$

$$k = \frac{\log(N-1)}{\log(m-1)} \tag{5}$$

valid values: $m > 2$ and $N > 2$.

For Matlab/Octave,

```
a=1:ceil(log(N-1)/log(m-1)); M_max=N*sum((m-1).^(a-1))
```

Note: use of "ceil" gives worse case scenario in which there are empty switch ports. This equation is accurate when Eq. 5 is an integer (then "ceil" is not used).

Solution 2: the minimum number of switches needed can be found by daisy-chaining switches together. We can enumerate this by seeing how many compute nodes can be attached to 1 switch, then 2 switches, and so on.

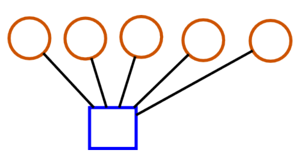


Figure 10: $m = 5$, $M = 1$, $N = 5$

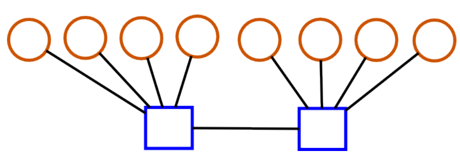


Figure 11: $m = 5$, $M = 2$, $N = 4 + 4 = 8$

Thus, for $M > 1$,

$$N = (m-1) + (M-2)(m-2) + (m-1) = 2(m-1) + (M-2)(m-2) \tag{6}$$

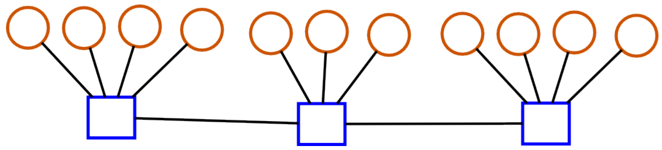


Figure 12: $m = 5$, $M = 3$, $N = 4 + 3 + 4$

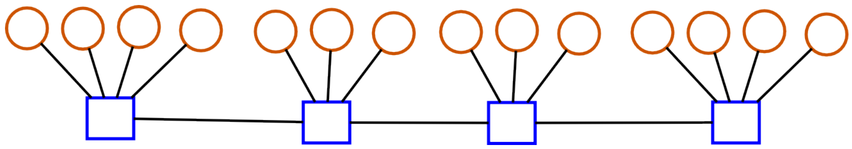


Figure 13: $m = 5$, $M = 4$, $N = 4 + 3 + 3 + 4$

This solution has a high hop count and low bisection bandwidth of 1.

Solving for M,

$$\boxed{M_{min} = \frac{N - 2(m-1)}{m-2} + 2} \tag{7}$$

where $m > 2$ and $N > 2(m - 1)$.

For Matlab/Octave,

```
M_min=(N-(2*(m-1)))/(m-2)+2
```

As an example, when $N = 1000$ and $m = 24$, the maximum number of switches is 553,000 (solution 1) and the minimum is 46 (solution 2).

6.2 more than one port per compute node

What is the maximum switch count when there are two ports per compute node ($n = 2$)?

Solution 1 (maximum number of switches): Start with one compute node and a switch with m ports (for this example let $m = 5$). At this tree level ($k = 1$) we have $m = 5$, $M' = 2$, and $N - 1 = 2 * 4 = 8$. (M' refers to the number of switches for this one compute node).

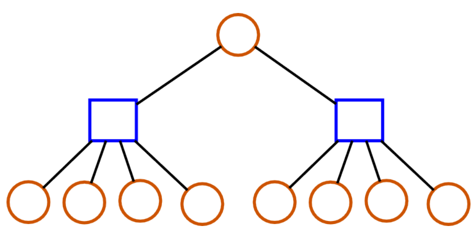


Figure 14: $m = 5$, $M' = 2$, $N - 1 = 2 * 4$ (tree level $k = 1$).

Generalizing, the number of nodes this one can connect to is

$$N = n(m-1)^k \tag{8}$$

and the number of switches for this one compute node is

$$M' = n \sum_{a=1}^{k} (m-1)^{(a-1)} \tag{9}$$

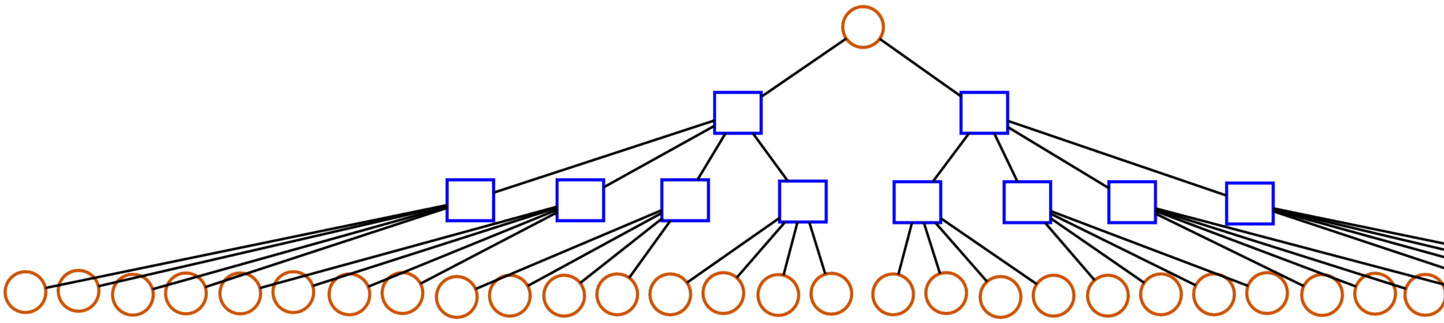


Figure 15: $m = 5$, $M' = 2 + 2 * 4$, $N - 1 = 2 * 4 * 4$ (tree level $k = 2$).

As before, the total number of switches needed is $M = NM'$,

$$\boxed{M_{max} = Nn \sum_{a=1}^{k} (m-1)^{(a-1)}} \tag{10}$$

and we can solve Eq. 8 to find k

$$\log(N-1) = \log(n) + k \log(m-1) \tag{11}$$

$$k = \frac{\log((N-1)/n)}{\log(m-1)} \tag{12}$$

For Matlab/Octave,

```
a=1:ceil(log((N-1)/n)/log(m-1)); M_max=N*n*sum((m-1).^(a-1))
```

Note: use of "ceil" gives worse case scenario in which there are empty switch ports. This equation is accurate when Eq. 12 is an integer (then "ceil" is not used).

The maximum number of switches as given by Eq. 10 is expected to be less than the value from Eq. 7 since there are more ports supplied at the compute node.

Solution 2 (minimum number of switches): Again we will start with $n = 2$ and find how many compute nodes are supported with 1 switch, then 2 switches, and so on. Here we will assume $m = 4$. We cannot use one switch for two compute nodes because it violates our earlier assumption that no compute node should connect to the same switch twice.

With two switches, we can connect 4 compute nodes.

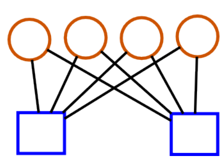


Figure 16: $m = 4$, $M = 2$, $N = 4$.

Clearly this becomes a mess to enumerate systematically. The important point is that the minimum number of switches is higher when $n > 1$. This is to be expected since there are more connections per compute node. Compare Fig. 6.2, 6.2 and Fig. 6.2, 6.2

As an example using the same parameters as before, when $N = 1000$ and $m = 24$, the maximum number of

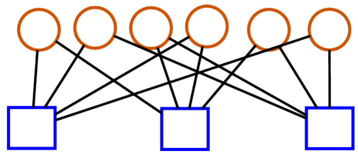


Figure 17: $m = 4$, $M = 3$, $N = 6$.

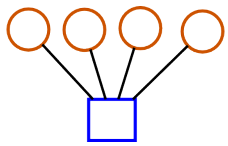


Figure 18: $m = 4$, $M = 1$, $N = 4$.

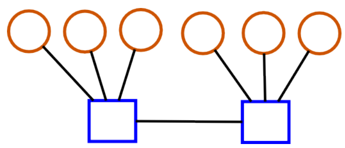


Figure 19: $m = 4$, $M = 2$, $N = 6$.

switches is now 72,000 (solution 1) and the minimum is greater than the minimum found when $n = 1$. The maximum was expected to decrease (compared to the maximum for $n = 1$) since there are more ports at the compute node.

As a check, the difference between the maximum and minimum number of switches should go to zero as n approaches $N - 1$.

```
N=1000; m=4;
for n=1:(N-1),
  a=1:ceil(log((N-1)./n)./log(m-1));
  M_max(n)=N*n.*sum((m-1).^(a-1));
end
n=1:(N-1);
plot(n,M_max)
figure; plot(1:N-2,M_max,"o","markersize",15)
xlabel("number of ports per node"); ylabel("maximum number of switches needed"); title("1000 compute no
print -dpng maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
print -deps maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
figure; semilogx(1:N-2,M_max,"o","markersize",15)
xlabel("number of ports per node"); ylabel("maximum number of switches needed"); title("1000 compute no
print -dpng maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
print -deps maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
figure; loglog(1:N-2,M_max,"o","markersize",15)
xlabel("number of ports per node"); ylabel("maximum number of switches needed"); title("1000 compute no
print -dpng maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
print -deps maximum_number_of_switches_needed_versus_compute_node_port_count_for_1000_compute_nodes_and
```

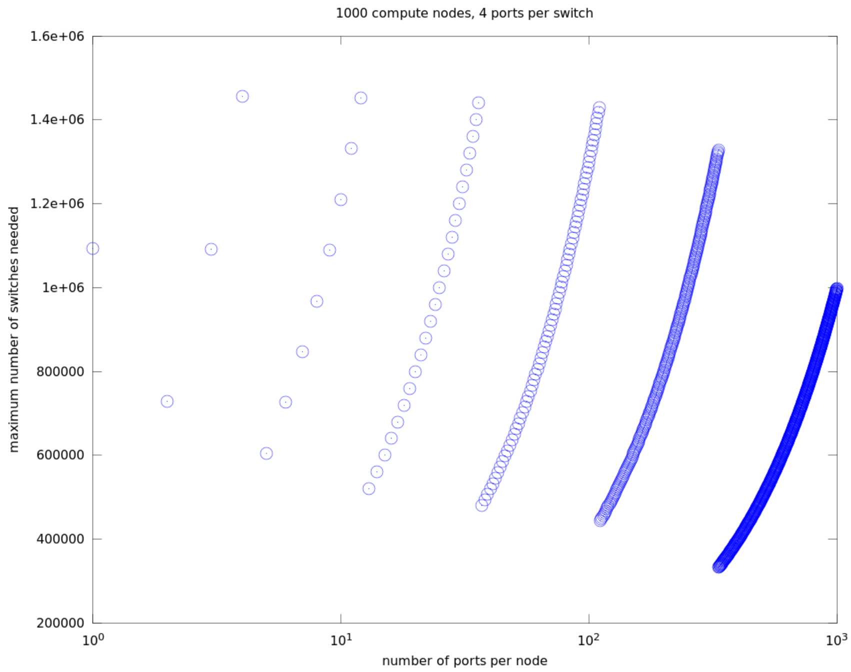


Figure 20: $m = 4$, $M = 3$, $N = 6$.

7 network alterations

7.1 swap computer connections

1. pick random pair from connections database [cA, s1]
2. pick random pair from connections database [cB, s2]
3. check that (cA != cB) and (s1 != s2)
4. check that the suggested rearrange "[cA, s1] and [cB, s2] transform to [cA, s2] and [cB, s1] in connections database" doesn't create an invalid network (i.e., more than one of the compute ports plugged into the same switch)
5. rearrange: [cA, s1] and [cB, s2] transform to [cA, s2] and [cB, s1] in connections database

Note: we actually could let the evolution include "invalid" networks (i.e., multiple computer ports plugged into same switch). However, this would probably result in a longer evolution (if the optimal network doesn't include this redundancy).

7.2 swap switch connections

1. pick random pair from connections database [s1, X]
2. pick random pair from connections database [s2, Y]
3. check that (X != Y) and (s1 != s2)
4. check that the suggested rearrange "[s1, X] and [s2, Y] transform to [s2, X] and [s1, Y] in connections database" doesn't create an invalid network (i.e., more than one of the compute ports plugged into the same switch)

5. rearrange: [s1, X] and [s2, Y] transform to [s1, Y] and [s2, X] in connections database

It appears that "swap switch connections" and "swap computer connections" have sufficient overlap that they can be merged to one routine.

7.3 swapping sets of connections on switches

1. pick random pair from connections database [s1, X1]
2. pick random pair from connections database with same switch, [s1, X2]
3. pick random pair from connections database [s2, Y1]
4. pick random pair from connections database with same switch, [s1, Y2]
5. check that (s1 != s2) and (X1 != Y1) and (X1 != Y2) and (X2 != Y1)
   Note that we don't have to check either (X1 != X2) or (Y1 != Y2) as this is done when the initial valid network is created
6. check that the suggested rearrange doesn't create an invalid network
7. rearrange: [s1, X1], [s1, X2], [s2, Y1], [s2, Y2] transform to [s1, Y1], [s1, Y2], [s2, X1], [s2, X2] in connections database

Q: is swapping multiple ports between two switches distinct from performing the same number of swaps on individual port pairs on switches?
For example,
"[s1, X1], [s1, X2], [s2, Y1], [s2, Y2] transform to [s1, Y1], [s1, Y2], [s2, X1], [s2, X2]" is equivalent to
"[s1, X1], [s2, Y1] transforms to [s1, Y1], [s2, X1]
and
[s1, X2], [s2, Y2] transforms to [s1, Y2], [s2, X2]"

8 Fitness functions

8.1 bisection bandwidth

Page 52 of the thesis "A complexity theory for VLSI" by C. Thompson (1980) defines bisection bandwidth for communication graphs as half the compute nodes being separated.

How to measure bisection bandwidth: chose a random set of half the compute nodes. Determine how many cuts need to be made to separate the two halves. There are many possible combinations of switches, and which nodes are in which half also matters.

Question posted to Networkx Google Group

Question: how to halve the compute nodes?

Simplification: consolidate compute nodes which are on the same switch. This only applies to compute nodes with one port.

Idea: use strong repulsion between the compute nodes?

Does recording the paths between compute nodes help?

8.1.1 number of bisections

How many times does the bisection need to be measured?

To answer this, we need to know how many unique permutations there are of the network. Suppose there are N computers and M routers. Then the number of halves of computers is $N(N-1)(N-2)\cdots(N/2)$:

$$\prod_{x=0}^{N/2}(N-x) \tag{13}$$

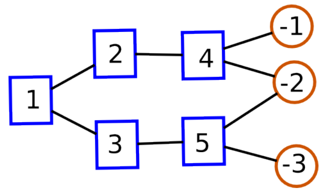


Figure 21: $m = 1, 2$, $M = 5$, $N = 3$. The hop count between -1 and -3 is 6, not 4.

Similarly, the number of (unequal) router partitions is $M!$. Thus the number of unique network partitions, U, is

$$U = M! \prod_{x=0}^{N/2} (N - x) \tag{14}$$

If we want to find the minimum bisection, then an (orderly) exhaustive search is necessary. Instead, do a random search (random swaps of edges) and have some confidence of having found the minimum.

Given U items, how many picks p are needed to get a specific item with a certain confidence level q. Constraints: (1) the search p doesn't stop when the item is found, (2) picks are made with replacement – there are always U items.

As an example, suppose $U = 2$. Then $p = 1$ gives $q = 50\%$. The confidence increases when p increases, but it never reaches 100%. For $p = 2$, the outcomes in which we find the specific item are ((p_1 and p_2) or (p_1 and !p_2) or (!p_1 and p_2)). Thus $p = 2$ yeilds $q = .5 * .5 + .5 * .5 + .5 * .5$ =75%. An easier way of expressing this is to ask the opposite: which outcomes are not going to yield the specific item? When we don't pick it in all tries: (!p_1 and !p_2). Then we care about 1-(.5*.5)=75%.

One more pick, $p = 3$. The cases in which we find a specific item are the same as not finding it after 3 tries: $1 - (!p_1$ and $!p_2$ and $!p_3) = 1 - (.5 * .5 * .5)$ =87.5%.

Since $p_i = 1/U$, then $!p_i = 1 - (1/U)$.

Once we can find the confidence based on p picks, we can invert the relation and calculate the number of picks needed to have a certain confidence level. Given p picks of U items, the confidence C that the specific item was picked at least once is

$$C = 1 - \left(1 - \frac{1}{U}\right)^p \tag{15}$$

Invert this to find $p(C, U)$

$$\left(1 - \frac{1}{U}\right)^p = 1 - C \tag{16}$$

$$p = \frac{\log(1 - c)}{\log(1 - (1/U))} \tag{17}$$

As an example, suppose we want 90% confidence ($C = 0.9$) and there are 100 items to pick from ($U = 100$). Then

$$p = \lceil \frac{\log(1 - .9)}{\log(1 - (1/100))} \rceil = 230 \tag{18}$$

We would need 230 picks.

8.2 hop count

http://en.wikipedia.org/wiki/Shortest_path_problem

Problem: Given connections

$$[[-1, 4], [-2, 4], [-2, 5], [-3, 5], [3, 5], [1, 3], [1, 2], [2, 4]] \tag{19}$$

See Fig. 8.2. Routes between compute nodes can only use positive nodes.

Objective: seek the shortest path between two negative nodes which contain no negative nodes.

Method: for $A < 0$ and $B < 0$, create a new graph with nodes A, B, and all positive nodes. Then find the shortest path between A and B. Repeat for all compute node pairs.

8.3 single-source shortest path

http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

8.3.1 All pairs shortest Path

- http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
- http://www.cs.rochester.edu/u/nelson/courses/csc_173/graphs/apsp.html
- http://stackoverflow.com/questions/5249857/all-pairs-all-paths-on-a-graph

Might be able to get away with finding average hop count and maximum hop count if (1) that is all we care about and (2) there's a faster algorithm for those counts.

8.4 Multi-objective optimization

When a fitness function has multiple components, i.e.,

$$f = a(\text{cost}) + b(\text{latency}) + c(\text{bisection bandwidth}) \tag{20}$$

Then how do you chose values for a, b, and c? The outcome of the evolution is expected to be sensitive to changes in each.

9 problems

10 future task list

Once the hop counter is implemented, it would be useful to validate metrics against analytic values for mesh, torus, fat tree topologies.