Wireless Communications: Principle and Applications
Project Report:
Task Allocation in Distributed Service Networks

Xiao Feng, 5100309117, Wei Fan, 5100309131

BACKGROUND

Distributed service network(DSN) is a new generation distributed core network architecture proposed by China Mobile Research Institute。It has characteristics of the telecommunication services and mobile Internet services such as fast, flexible, low cost and scalability.

DSN achieve traditional telecommunication network’s core functions in a way of P2P. In addition, it can also provide the ability of sustained operations, adaptive load balancing, distributed storage, dynamic resource scheduling and other new networking capabilities, which on the one hand is to better support mobile Internet services, on the other hand is able to achieve a lower cost telecommunications services.

And fully distributed DNS shown as Figure 1. It’s functions spread to access network node and external node, which will make full use of the access network nodes’ and terminal nodes’ computing power, storage capacity and bandwidth resources. Thanks to the distributed technologies like P2P, the DSN will own many features that traditional client-server system don’t have, such as distribution, homogenization, robustness, self-organization, Intelligent routing and High cost performance. And now, the research scope of DSN is var-

Figure 1: fully distributed DNS

ious. There are some typical topics such as Internet business characteristics and the implementation mechanism, DSN architecture research, Homogenization of Super Node - Node architecture and P2P based research.

OUR TOPIC

A simple schematic diagram of DNS shown as Figure 2. In the center is a super node, it can control other DSN node. Note that every DSN node has the same function, just different in performance. And every DSN node is connected to some tasks. At the same time, each task can be processed on any DSN node. A task can be processed on the nearest node. But maybe at that moment another node is idle and have a better performance than the nearest node. So the task may be transmit to another node and processed in that node. After the task is done, the result will be transmit back to the nearest node. And the content of our research is how to allocate all tasks in fully distributed DSN to make the sum of whole processing time and transmitting time to a minimum.

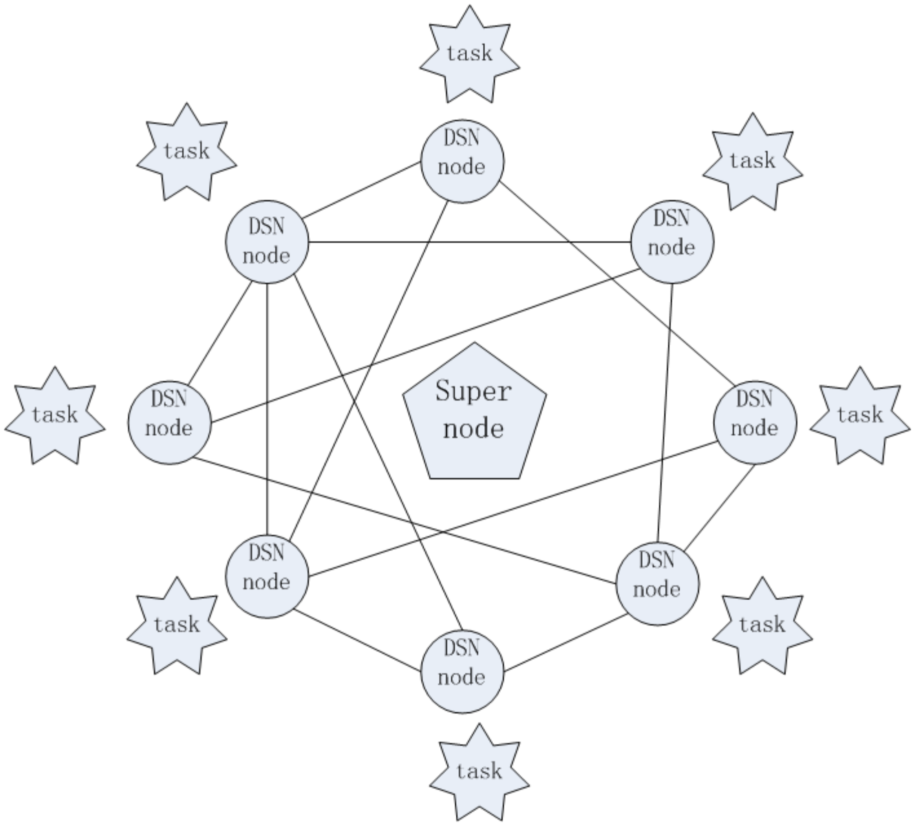

Figure 2: schematic diagram of DNS

SIMPLE MODEL

N tasks

M DSN nodes

O_i the agent of task i directly connects with node O_i

L_i work load of task i

S_i processing speed of node for each task i when it runs k tasks

m_{ij} transport delay(time) between node i and j

b_i the biggest number of tasks on node i

$$f_{ij} = \begin{cases} 1 & \text{task } i \text{ runs on node } j \\ 0 & \text{other} \end{cases}$$

a_i task i runs on node a_i,

$$a_i = \sum_{j=1}^{N} f_{ij} \cdot j$$

n_i current number of tasks on node i, $0 \leq n_i \leq b_i$,

$$n_i = \sum_{j=1}^{N} f_{ji}$$

AIM:minimize total completion time

$$min \ \sum_{i=1}^{N} (m_{a_i O_i} + \frac{L_i}{S_{a_i n_{a_i}}})$$

$$s.t.$$

$$\forall i, n_i \leq b_i$$

$$\sum_{i=1}^{M} n_i = N$$

$$\forall task \ i, \sum_{j=1}^{M} f_{ij} = 1$$

Solve:Dynamic Programming

$F(i, n_1, n_2 ... n_{M-1})$ represents the minimum total completion time among the first i tasks, where node i runs n_1 tasks.... $n_M = i - n_1 - n_2 ... n_{M-1}$

mapping vector$(n_1, n_2 ... n_{M-1})$ to status delegate P, thus,$F(i, n_1, n_2 ... n_{M-1})$ could be represented by$F(i, P)$

$$F(i, P) = min\{F(i-1, P') + m_{jO_i} + \frac{L_i}{S_{a_j j}}\}, \quad 1 \leq j \leq M$$

using $A_{i,P}$ to store corresponding strategy,under P situation, i is best running on node A_{iP}

SIMPLE MODEL + BALANCED LOAD

Algorithm 1 minimum total completion time

Require: **Input**: $O_i, S_{ij}, L_i, b_i, m_{ij}$;

Ensure: **Output**:f_{ij}, a_i, n_i;

$\forall i, \forall P, \quad F(i,P) \leftarrow \infty$;
F(0,0) $\leftarrow$ 0;
for (i $\leftarrow$ 1 to N) **do**
  for (P $\leftarrow$ 0 to P_max) **do**
    if (P is legal) **then**
      for (j $\leftarrow$ 1 to M) **do**
        if $(n_j \le b_i \&\& F(i,P) < F(i-1,P') + m_{jO_i} + \frac{L_i}{S_{a_j j}})$ **then**
          $F(i,P) \leftarrow F(i-1,P') + m_{jO_i} + \frac{L_i}{S_{a_j j}}$;
          $A(i,P) \leftarrow j$
        end if
      end for
    end if
  end for
end for
using $A(i,P)$ to calculate f_{ij}, a_i, n_i

Add constraints to the allocation of tasks, we could achieve this by setting deviation of work load on each node no bigger than a threshold value K,$i.e.$

$$\forall node\ i, \forall node\ j, \quad |\sum_{k=1}^{N} f_{ki}L_k - \sum_{k=1}^{N} f_{kj}L_k| \le K$$

then the problem changes to

$$min\ \sum_{i=1}^{N}(m_{a_i O_i} + \frac{L_i}{S_{a_i n_{a_i}}})$$

$$s.t.$$

$$\forall node\ i, n_i \le b_i$$
$$\sum_{i=1}^{M} n_i = N$$
$$\forall task\ i, \sum_{j=1}^{M} f_{ij} = 1$$
$$\forall node\ i, \forall node\ j, \quad |\sum_{k=1}^{N} f_{ki}L_k - \sum_{k=1}^{N} f_{kj}L_k| \le K$$

we could solve this problem by slightly modifying the algorithm.

Algorithm 2 minimum total completion time with balanced load

Require: **Input**: $O_i, S_{ij}, L_i, b_i, m_{ij}$;

Ensure: **Output**:f_{ij}, a_i, n_i;

$\forall i, \forall P, \quad F(i, P) \leftarrow \infty$;
F(0,0) $\leftarrow$ 0;
for (i $\leftarrow$ 1 to N) **do**
  for (P $\leftarrow$ 0 to P_max) **do**
    if (P is legal) **then**
      for (j $\leftarrow$ 1 to M) **do**
        if ($n_j \leq b_i \&\& F(i, P) < F(i-1, P') + m_{jO_i} + \frac{L_i}{S_{a_j j}}$) **then**
          Recursively compute current load status on each node using $A(i, P)$
          if (current load status + L_i on node j is legal) **then**
            $F(i, P) \leftarrow F(i-1, P') + m_{jO_i} + \frac{L_i}{S_{a_j j}}$;
            $A(i, P) \leftarrow j$
          end if
        end if
      end for
    end if
  end for
end for
using $A(i, P)$ to calculate f_{ij}, a_i, n_i

Complicated Model

Remove the definition S_{ij}, L_i, b_i, n_i in Simple Model.

Add:

Each task i has a resource demand vector $\overrightarrow{D_i} = (d_{i1}, d_{i2}...d_{ik})$, k, the number of total types of resources.

Each DSN node i has a resource vector $\overrightarrow{R_i} = (r_{i1}, r_{i2}...r_{ik})$

t_{ij} time needed on node i to finish task j

Problem:

$$min \sum_{i=1}^{N} (m_{a_i O_i} + t_{a_i i})$$

$$s.t.$$

$$\forall r_{ij}, r_{ij} \geq \sum_{k=1}^{N} f_{ki} d_{kj}$$

$$\forall task\ i, \sum_{j=1}^{M} f_{ij} = 1$$

This is an NP-Hard problem, which can not be solved by dynamic programming.The time complexity could be represented by $O(M^N)$. However, noticing that if t_{ij} is not quite different with each other, we could simplify the problem by setting that we only allocate a task i among the first A nearest DSN nodes to O_i,$i.e.$,the first A nodes which has the smallest $m_{O_{ij}}$.

Then the time complexity could be reduced to $O(A^N)$, $(\frac{A}{M})^N$ times of before.We use backtracking algorithm to solve this problem.

Algorithm 3 Complicated Model:minimum total completion time

Require: **Input**: $O_i, \overrightarrow{R_i}, \overrightarrow{D_i}, t_{ij}, m_{ij}$;

Ensure: **Output**:f_{ij}, a_i;

PROC Backtracking(step,nowcost,$\overrightarrow{R}$)
if (nowcost≥best) **then**
  Return;
end if
if (step=n+1) **then**
  best=nowcost;
  update best strategy f_{ij}, a_i
end if
for (each possible node j) **do**
  if ($\overrightarrow{R}$allows node j to run task *step*) **then**
    backtracking(step+1,$nowcost + t_{step,j} + m_{O_{step},j}, \overrightarrow{R} - \overrightarrow{D_{step}}$);
  end if
end for
ENDPROC
$best \leftarrow \infty$;
backtracking(1,0,$\overrightarrow{R}$);