Load Sharing in Networking Systems

Lukas Kencl
Intel Research Cambridge
MFF UK Praha, November 2005

(With lots of help from Weiguang Shi, University of Alberta, and citations from D. Thaler, R. Ravishankar and K. Ross)
Outline

- Intel Research Cambridge
- Load sharing in networking systems
  - Problem statement
  - Motivation
  - Imbalance due to traffic properties

................................. Break ..........................................................

- Inspiration: distributed Web caching
- Solutions and properties:
  - Adaptive HRW hashing
  - Adaptive Burst Shifting

................................. Break ..........................................................

- Seminar:
  - Q&A, Exercises and Homeworks
  - IR Movies!!!
Intel R&D Commitment

Intel Research: 4 Labs, target 80 people worldwide

Over 75 labs & 7000 R&D professionals worldwide

www.intel.com/research
Intel Research Mission

- Build the technical leadership, knowledge assets and systems perspective to make Intel a preeminent driver of disruptive information technologies

David Tennenhouse
Vice President, Corporate Technology Group
Director of Research, Intel Corporation
What is unique about Intel Research?

- Research is largely exploratory
  - Off roadmap, 7-15 years out on timeline
- Innovative collaborative model engaging with key Universities
  - “Lablets” (Berkeley, Seattle, Pittsburgh, Cambridge) – Intel employee facilities co-located on University campus to facilitate collaboration
  - Open Collaborative Agreements signed with Universities
- Strategic Research Projects (SRP) inside Intel
  - Technology transfer mechanism from open collaborative research
  - Proprietary exploratory research outside scope of any business unit
- Alignment of University engagements
  - Research Council, Academic Relations, Labs, Visiting Faculty, Internships!!!
**Intel Research Cambridge (IRC)**

- Established in March 2003
- Currently 12 full-time researchers
- Visiting faculty, student interns
- Make sure the smartest work with us: cooperation with Cambridge University, as well as others throughout UK, Europe and elsewhere
IRC Key Research Areas

- Optical Packet Switching
- Virtualisation – *Xen*
  - Para-virtualisation - run modified or new OS on specialised guest architecture
- Wired Networking
  - *CoMo* – Continuous Monitoring
  - Adaptive methods
- Anticipate the wireless
  - WIP: from internet to Intairnet
  - wireless access network using multidirectional antennas
  - HAGGLE: mobile ad-hoc networks
- Ubiquitous computing
Load Sharing in Networking Systems

- Problem Statement
Map packets to processors

Assumptions:
• Data arrives in packets.
• Any processor can process any packet.
• Heterogenous processor capacity $\mu_j$.

Task:
• Map packets to processors so that load within some measure of balance.
Packet flows: avoid remapping or out-of-sequence!!!

Assumptions:
• Data arrives in packetized flows.
• Any processor can process any packet.
• Heterogenous processor capacity $\mu_j$.

Task:
• Load on processors within some measure of balance.
• Same flow to same processor (reordering, context).

Advantage: system optimization.
Drawback: complexity, overhead.
Reduce state maintenance of Flow-to-Processor Mapping

Upon packet arrival, a decision is made where to process the packet, based on the flow identifier. A *flow-to-processor mapping* $f$ is thus established.
Acceptable load sharing as a measure of balance

- Processing load on processor $j$: $\lambda_j(t)$
- Capacity on processor $j$: $\mu_j$
- Workload intensity on processor $j$: $\rho_j(t) = \frac{\lambda_j(t)}{\mu_j}$
- Total system workload intensity: $\rho(t) = \sum \frac{\lambda_j(t)}{\sum \mu_j}$

Acceptable load sharing:
- if $\rho(t) \leq 1$ then $\forall j$, $\rho_j(t) \leq 1$,
- if $\rho(t) > 1$ then $\forall j$, $\rho_j(t) > 1$.

"No single processor is overutilized if the system in total is not overutilized, and vice versa."
Coefficient of Variation (CV) as a measure of balance

\[
CV = \frac{\text{standard deviation}}{\text{mean}} = \frac{\sqrt{\text{Var}(x)}}{E[x]}
\]

\[
(CV)^2 = \frac{\text{Var}(x)}{E[x]^2}
\]

Useful, as it takes the scale of measurements out of consideration.
Minimizing Disruption

Possible Goal: Acceptable load sharing without maintaining flow state information and yet minimizing the probability of mapping disruption (flow remapping or reordering).

Special 0-1 Integer Programming Problem:

\[
\max \sum_v \Delta_v(t) \cdot \sum_j (1\{f(t-\Delta t)(v) = j\} \cdot 1\{f(t)(v) = j\}),
\]

while \( \sum_v a_v(t) \cdot 1\{f(t)(v) = j\} = \lambda_j(t) \leq \mu_j, \forall j. \)

\( v \) - flow identifier vector in the packet header,
\( f(t)(v) \) - function mapping flows to processors, changing over time,
\( \Delta_v(t) \in \{0, 1\} \) - indicator if \( v \) has appeared in the intervals \((t-2\Delta t, t-\Delta t)\) and \((t-\Delta t, t)\),
\( a_v(t) \) - how many times has \( v \) appeared in the interval \((t-\Delta t, t)\),
\( \Delta t \) - iteration interval.

We suspect an \( NP \)-complete problem – use **heuristics**.
Summarize

- Balance load
- Avoid remapping - or packets out-of-sequence
- Minimize overhead
Load Sharing in Networking Systems

- Motivation
Practical Examples

- Distributed Router, or Parallel Forwarding Engine
- Server Farm Load Balancer
- Network Processor
Network Processor

- Network processor:
  - Pool of multi-threaded forwarding engines
  - Integrated or attached GPP
  - Memories of varying bandwidth and capacity
- Large parallel programming problem:
  - Concurrency
  - Packet order
  - Resource allocation
  - Load balancing
- At 10 Gbps, packets arrive every 36 ns!!!
Load Sharing in Networking Systems

• Traffic properties make balancing difficult
Traffic Properties

- Address distribution – 32bit address space very unevenly assigned and populated
- Burstiness
- Power law
TCP Traffic is Bursty

TCP behaviour: ideal and typical

Packets arrive in periodic bursts!
Networking Traffic Exhibits Power-Law (Zipf-law) Properties

Flow popularities in traffic traces

\[ P(R) \sim 1/R^a \]

Frequency of an event as a function of its rank obeys power-law. Popularity of network flows with \( a \) close to 1 (from above).
Power-law leads to imbalance if static mapping (Shi et al, 2004)

\( m \) - number of processors,
\( K \) - number of distinct object addresses or flow IDs,
\( p_i (0 < i < K) \) - popularity of object \( i \),
\( q_j (0 < j < m) \) - number of distinct addresses or flow IDs mapped to processor \( j \).

Given that the average popularity of the \( K \) objects, \( E[p_i] \), is \( \frac{1}{K} \), we have

\[
CV[p_i]^2 = \frac{Var(p_i)}{E[p_i]^2} = \frac{E[p_i^2] - E[p_i]^2}{E[p_i]^2} \\
= \frac{1}{K} \sum_{i=1}^{K} \frac{1}{Z^2(i)^{2-2\alpha}} - 1 \\
= \frac{K}{Z^2} \sum_{i=1}^{K} i^{-2\alpha} - 1
\]

Substituting the \( CV[p_i]^2 \) in Eq. 2, we have

\[
CV[q_j]^2 \sim \frac{K(m - 1)}{Z^2(K - 1)} \sum_{i=1}^{K} i^{-2\alpha}
\]

As \( \alpha > 1 \) and \( K \to \infty \), items \( Z \) and \( \sum_{i=1}^{K} i^{-2\alpha} \) converge, and thus \( CV[q_j]^2 \) is non-zero.

Zipf-like distributions (Eq. 1) are known to have infinite variance when \( \alpha \leq 3 \) and infinite mean when \( \alpha \leq 2 \). This is the reason that a hash based scheme, such as HRW [5], is not able to achieve load balancing when the population distribution of objects in its input space, in our case destination IP addresses, is Zipf-like with \( \alpha > 1 \).
Conclusion Part I

- Complex problem
- Traffic properties work against static solutions
Break 5 minutes
Load Sharing in Networking Systems

- Inspiration: Distributed Web Caching and Web Server Load Balancing
What is a Distributed Web Cache

Proxy Cache stores Web objects locally
Collection of distributed Web caches

Intranet
Internet
**HRW Mapping**

Highest Random Weight (HRW) Mapping,


**Def.: Object-to-Processor mapping function** \( f : V \rightarrow M : \)

\[ f(v) = j \iff x_j \cdot g(v, j) = \max_k x_k \cdot g(v, k), \]

where \( v \) is the object identifier vector, \( x = (x_1, \ldots, x_m) \) is a weights' vector and \( g(v, j) \in (0, 1) \) is a pseudorandom function of uniform distribution.

- **Minimal disruption of mapping** in case of processor addition or removal/failure.
- **Load balancing over heterogenous processors**: weights' vector \( x \) is in a 1-to-1 correspondence to \( p = (p_1, \ldots, p_m) \), the vector of object fractions received at each processor.
- **Pseudorandom function** \( g(v, j) \in (0, 1) \) can be implemented as a fast-computable hash function.

**Example:** 3 processors

Map to max of:

- \( x_1 \cdot g(v, 1) \)
- \( x_2 \cdot g(v, 2) \)
- \( x_3 \cdot g(v, 3) \)

<table>
<thead>
<tr>
<th>Processor</th>
<th>( x_1 \cdot g(v, 1) )</th>
<th>( x_2 \cdot g(v, 2) )</th>
<th>( x_3 \cdot g(v, 3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ 0 \quad 1 \quad 2 \quad 3 \]
HRW Minimal Disruption

Minimal disruption of mapping in case of processor addition:

Example: Add processor No. 4, vectors mapped either (i) as before addition or (ii) to the newly added processor – minimal number of vectors change mapping.

Partitioning into contiguous set

HRW
HRW Load Balancing

$m$ - number of servers, $S_1, ..., S_m$

$K$ - number of distinct object addresses or flow IDs, from the set of objects $O$

$p_i (0 < i < K)$ - popularity of object $i$,

$q_j (0 < j < m)$ - number of distinct addresses or flow IDs mapped to processor $j$, e.g. sum of the popularities of objects mapped to $S_j$

Let the mapping $M$ assign objects $o_{i1}, o_{i2}, \ldots, o_{ik}$ to server $S_i$. Since $M$ assigns all objects to servers, we can view $o_{i1}, o_{i2}, \ldots, o_{ik}$ as a selection of size $k$ from the set $O$, such that any object $o_i$ is selected with the probability $1/K$. We can similarly model server popularities by viewing $p_1, p_2, \ldots, p_K$ as a sample of size $k$ from $p_1, p_2, \ldots, p_K$, taken without replacement. In this case, the population mean $\overline{p} = (p_1 + p_2 + \ldots + p_K)/K = 1/K$, since the $p_i$ sum to 1. Let the population variance be $\sigma_p^2$.

**Theorem 4 (Hash-Allocation Request Balancing):** Let $K$ objects be partitioned among $m$ servers using HRW, with each server receiving exactly $k = K/m$ objects. If $p_i, \overline{p}, \sigma_p^2$ and $q_i$ are as defined above, then the square of the coefficient of variation of $q_i$ is given by

$$CV[q_i]^2 = \left(\frac{m-1}{K-1}\right) CV[p]^2$$

and, hence, when $p$ has finite variance

$$lim_{K \to \infty} CV[q_i] = 0.$$
Load Sharing in Networking Systems

• Adaptive Methods
  - Adaptive HRW Hashing
  - Adaptive Burst Shifting
Adaptive HRW Hashing
(Kencl, Le Boudec 2002)
Adaptive HRW mapping

Compute mapping function

$$FE = f(fID, x)$$

Incoming packet

LB

Multiple Processing Units (FEi)

(Weights, ~ Fraction of traffic)

1. $(x_1, p_1)$
2. $(x_2, p_2)$
3. $(x_3, p_3)$
4. $(x_4, p_4)$
N. $(x_N, p_N)$
Adaptation through Feedback

**Problem:** incoming requests are packets, not flows! Packets not evenly distributed over flows

disproportionately distributed over the request object space

HRW mapping not sufficient for acceptable load sharing bounds

need to adapt!

1. Filtered workload intensity $(j(t)) = \rho_j(t)$
2. Evaluate
   \[ \rho(t) = (\rho_1(t), \rho_2(t), \ldots, \rho_m(t)) \]
   (compare threshold).
3. Compute new
   \[ x(t) = (x_1(t), \ldots, x_m(t)) \]
4. Download new
   \[ x := x(t) \]

- Trigger definition targets *preventing overload*, if system in total not overloaded, and vice versa.
- A *threshold* triggers adaptation when close to load sharing bounds.
- Flow-to-processor mapping $f$ becomes a *function of time* $f(t)$.
- Adaptation may cause *flow remapping*! How to minimize the amount remapped?
Adaptation Algorithm

Start

Wait time $\Delta t$

Compute filtered processor workload intensity $\rho(t)$

Yes

Adapt weights' vector $x$ and upload

Adaptation Policy

No

Trigger Adaptation?

Triggering Policy
Triggering Policy

Dynamic workload intensity threshold
\[ \varepsilon'_\rho(t) = \frac{1}{2} (1 + \rho(t)) \]

**Example:**
\[ \rho(t) = (0.8, 0.2, 0.2), \rho(t) = 0.4 \]
\[ \Rightarrow \varepsilon'_\rho(t) = 0.7, \]
\[ \rho_1(t) > \varepsilon'_\rho(t) \Rightarrow \text{adapt} \]

Triggering threshold
\[ \varepsilon_\rho(t) = \max(\varepsilon'_\rho(t), \text{upper}) \]
or vice versa

Hysteresis bound
upper: \((1 + \varepsilon_H(t)) \cdot \rho(t)\)
lower: \((1 - \varepsilon_H(t)) \cdot \rho(t)\)

Triggering policy
(i) if \( \rho(t) \leq 1 \) and \( \max \rho_j(t) > \varepsilon_\rho(t) \)
then adapt;
(ii) if \( \rho(t) > 1 \) and \( \min \rho_j(t) < \varepsilon_\rho(t) \)
then adapt.
Adaptation Policy: Minimal Disruption

- $A$, $B$ - mutually exclusive subsets of $M=\{1,\ldots,m\}$, $M=A \cup B$.
- $\alpha \in (0, 1)$.
- $f$, $f'$ - two HRW mappings with the weights' vectors $x, x'$:
  
  $x'_j = \alpha \cdot x_j$, $\forall j \in A$, 
  
  $x'_j = x'_j$, $\forall j \in B$.

- $p_j, p'_j$ - fraction of objects mapped to node $j$ using $f$, $f'$.

1) $p'_j \leq p_j$, $j \in A$, 
   
   $p'_j \geq p_j$, $j \in B$.

2) Fraction of objects mapped to a different node by each mapping is **MINIMAL**, that is, equal to $|p'_j - p_j| \cdot |V|$ at every node $j$. 
Adaptation Policy: Minimal Disruption Example (3 proc.)

- reduced receives less, unaltered receives more, if reduction by a single, invariable multiplier.
- minimal disruption of the mapping.
Adaptation Policy

Let $\rho(t) \leq 1$. Then:

\[ x_j(t) : = c(t) \cdot x_j(t-\Delta t), \quad \text{if } \rho_j(t) > \varepsilon_p(t) \quad (j \text{ exceeds threshold } \varepsilon_p(t)), \]
\[ x_j(t) : = x_j(t-\Delta t), \quad \text{if } \rho_j(t) \leq \varepsilon_p(t) \quad (j \text{ does not exceed threshold } \varepsilon_p(t)). \]

If $\rho(t) > 1$, the adaptation is carried out in a symmetrical manner.

The weights' multiplier coefficient $c(t)$:

\[ c(t) = \left( \frac{\varepsilon_p(t)}{\min \{ \rho_j(t) \mid \rho_j(t) > \varepsilon_p(t) \}} \right)^{1/m} \]

Factor $c(t)$ is proportional to the minimal error and to the number of nodes.
Adaptive Burst Shifting
(Shi, MacGregor, Gburzynski 2005)
Adaptive Burst Shifting

Insight: the periodicity of bursts may allow shifting flows in-between bursts, without affecting order within flows.

The sufficient condition for in-order delivery: $T_j - T_i > L/u$, where $u$ is the processing rate (pkt/sec) of an FE.
Adaptive Burst Shifting

Burst Distributor Algorithm

STEP 2. IF a valid entry is found
STEP 3. THEN return the FE field of the entry
STEP 4. IF the table is not full
STEP 5. THEN return the index of the minimum-loaded FE
STEP 6. Return an invalid FE index
Performance evaluation
Adaptive HRW on generated traffic
Expectations

- Workload intensity on individual processors close to that of the system in total (*acceptable load sharing*);
- Packet loss probability lowered (*acceptable load sharing*);
- Persistent flows (appearing in two consecutive iterations) seldom remapped (*minimize disruption*).
AHH Keeps Per-processor Workload Intensity Close to Ideal

Naive, no LS.

Static HRW

Adaptive HRW

Max and min of all.
Packet Loss Significantly Reduced with the Adaptive Control Loop

Packet loss in excess of Ideal: Static, Adaptive.

Adaptive HRW saves on average 60% of packets dropped in by the static load sharing.
Minimal Disruption Property Ensures Few Flow Remappings

Adaptive control loop leads on average to:
- less than 0.05% of the appearing flows remapped per iteration;
- less than 0.2% of the persistent flows remapped per iteration.

Flows, per iteration: appearing, persistent and remapped.
AHH and ABS on generated traffic
Adaptive HRW and Adaptive Burst Switching

Packet drop rates.

Packet reordering.

Packet remapping.
Conclusion

- ABS better in preventing packet drop
- AHH better in preventing remapping and reordering, although larger table would improve ABS
- ABS works on much smaller time-scale – can address packet bursts
- AHH converges to optimal allocation, but timescale too long for bursts
- Best solution probably hybrid – under investigation – must be careful to avoid conflicting feedback-control mechanisms
The End

- Thank you!
- Seminar @ 10:40
Seminar
Seminar agenda

- Q & A
- Exercises 1, 2, 3
- Internships @ IRC
- Homework
- Q & A II.
- Cinema?
Exercise 1: Mapping Disruption

Let there be a system with $M$ servers of the same capacity, and an $(M+1)$th server be added to the system. Compute the amount of mapping disruption with the:

a) Contiguous mapping

b) Mod M mapping
Exercise 2: Prove the x2p relationship of HRW mapping

**Theorem 1** Let \( p_1, \ldots, p_N \) be given target probabilities. Reorder the caches so that \( p_1 \leq \cdots \leq p_N \). Let

\[
x_1 = (Np_1)^{1/N}
\]

and let \( x_2, \ldots, x_N \) be calculated recursively as follows:

\[
x_n = \left[ \frac{(N-n+1)(p_n - p_{n-1})}{\prod_{i=1}^{n-1} x_i} + x_{n-1}^{N-n+1} \right]^{1/(N-n+1)}.
\]

Then the robust hash algorithm with multipliers \( x_1, \ldots, x_N \) will route the fraction \( p_n \) of URLs to the \( n \)th cache for \( n = 1, \ldots, N \).
Homework:

1. Prove the mindisruption property of Adaptive HRW hashing

2. Try and design your own adaptive load balancing mapping that a) reduces packet reordering within flows, or b) reduces flow remapping.
Q & A
Internships? Films?

Thank you!

lukas.kencl@intel.com
http://www.intel-research.net/cambridge/