CS533
Modeling and Performance Evaluation of Network and Computer Systems

Workload Characterization Techniques

(Chapter 6)

Workload Characterization Techniques

Speed, quality, price. Pick any two. – James M. Wallace

- Want to have repeatable workload so can compare systems under identical conditions
- Hard to do in real-user environment
- Instead
  - Study real-user environment
  - Observe key characteristics
  - Develop workload model

→ Workload Characterization

2

Terminology

- Assume system provides services
- Workload components – entities that make service requests
  - Applications: mail, editing, programming ..
  - Sites: workload at different organizations
  - User Sessions: complete user sessions from login to logout
- Workload parameters – used to model or characterize the workload
  - Ex: instructions, packet sizes, source or destination of packets, page reference pattern, …

3

Choosing Parameters

- Better to pick parameters that depend upon workload and not upon system
  - Ex: response time of email not good
    - Depends upon system
  - Ex: email size is good
    - Depends upon workload
- Several characteristics that are of interest
  - Arrival time, duration, quantity of resources demanded
    - Ex: network packet size
  - Have significant impact (exclude if little impact)
    - Ex: type of Ethernet card

4

Techniques for Workload Characterization

- Averaging
- Specifying dispersion
- Single-parameter histograms
- Multi-parameter histograms
- Principal-component analysis
- Markov models
- Clustering

5

Averaging

- Characterize workload with average
  - Ex: Average number of network hops
- Arithmetic mean may be inappropriate
  - Ex: average hops may be a fraction
  - Ex: data may be skewed
  - Specify with median, mode

6

Specifying Dispersion

- Variance and Standard deviation
- C.O.V. (what is this?)
- Min-Max Range
- 10- 90-percentiles
- SIQR (what is this?)
- If C.O.V. is 0, then mean is the same
- If C.O.V. is high, may want complete histogram (next)
  - Or divide into sub-components and average those that are similar only
  - Ex: average small and large packet sizes

Case Study (1 of 2)

- Resource demands for programs at 6 sites
- Average and C.O.V.

Data	Average	C.O.V.
CPU time	2.19 sec	40.23
Number of writes	8.20	53.59
Number of reads	22.64	26.65

- C.O.V. numbers are high!
  - Indicates one class for all apps not a good idea

Case Study (2 of 2)

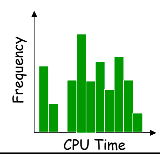
- Instead, divide into several classes
- Editing Sessions:

Data	Average	C.O.V.
CPU time	2.57 sec	3.54
Number of writes	19.74	4.33
Number of reads	37.77	3.73

Single-Parameter Histograms

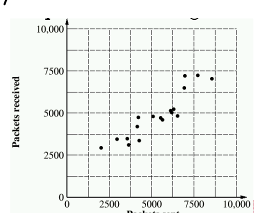
- Shows relative frequency of parameter values
- Divide into buckets. Values of buckets can be used to generate workloads
- Given n buckets, m parameters, k components nmk values
  - May be too much detail, so only use when variance is high

• Problem: may ignore correlation. Ex: short jobs have low CPU and I/O, but could pick low CPU high I/O



Multi-Parameter Histograms

- If correlation, should characterize in multi-parameter histogram
  - n-dimensional matrix, tough to graph n > 2
  - Often even more detailed than single parameter, so rarely used



Principal-Component Analysis

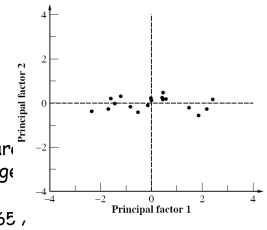
- Goal is to reduce number of factors
- PCA transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components

PCA: Example PCA (1 of 2)

- Jai, p. 78. Sending and Receiving packets
- Compute mean and standard deviation
- Normalize all points
  - Subtract mean, divide by standard dev
- Compute correlation (essentially, a line through the data that minimizes distance from line)
  - Principal component 1, rotate to be x axis
- Now, next, fit another line that also minimizes distance from line but is orthogonal (not correlated) to first line
  - Using eigenvalues and eigenvectors
  - Principal component 2, rotate to be y axis

13

PCA: Example PCA (2 of 2)



- Compute sums of squar explains the percentag of variation
  - Ex: Factor 1 → 32.565 / (32.565 + 1.435) or 96%
  - Ex: Factor 2 →1.435 / (32.565 + 1.435) or 4%

14

PCA: Example PCA (3 of 3)

Obs. No.	Variables x_s	Variables x_r	Normalized Variables x'_s	Normalized Variables x'_r	Principal Factors y_1	Principal Factors y_2
1	7718	7258	1.359	1.717	2.175	-0.253
2	6958	7232	0.922	1.698	1.853	-0.549
3	8551	7062	1.837	1.575	2.413	0.186
4	6924	6526	0.903	1.186	1.477	-0.200
5	6298	5251	0.543	0.262	0.570	0.199
6	6120	5158	0.441	0.195	0.450	0.174
7	6184	5051	0.478	0.117	0.421	0.255
8	6527	4850	0.675	-0.029	0.457	0.497
9	5081	4825	-0.156	-0.047	-0.143	-0.077
10	4216	4762	-0.652	-0.092	-0.527	-0.396
11	5532	4750	0.103	-0.101	0.002	0.145
12	5638	4620	0.164	-0.195	-0.022	0.254
13	4147	4229	-0.692	-0.479	-0.828	-0.151
14	3562	3497	-1.028	-1.009	-1.441	-0.013
15	2955	3480	-1.377	-1.022	-1.696	-0.251
16	4261	3392	-0.627	-1.085	-1.211	0.324
17	3644	3120	-0.981	-1.283	-1.601	0.213
18	2020	2946	-1.914	-1.409	-2.349	-0.357
$\sum x$	96336	88009	0.000	0.000	0.000	0.000
$\sum x^2$	567119488	462661024	17.000	17.000	32.565	1.435
Mean	5352.0	4889.4	0.000	0.000	0.000	0.000
Std. Dev.	1741.0	1379.5	1.000	1.000	1.384	0.290

- Use y1 for high load
- Not much gained in y2

15

Markov Models (1 of 2)

- Sometimes, important not to just have number of each type of request but also order
- If next request depends upon previous request, then can use Markov model
  - Actually, more general. If next state depends only on current state
- Ex: process between CPU, disk, terminal:

From/To	CPU	Disk	Terminal
CPU	0.6	0.3	0.1
Disk	0.9	0	0.1
Terminal	1	0	0

(Draw diagram Fig 6.4)

16

Markov Models (2 of 2)

- Also for application transitions
  - Ex: users run editors, compilers, linkers → markov model to characterize probability of type j after type I
- Also for page reference locality
  - Ex: probability of referencing page (or procedure) i after page (or proc.) j
- Really talks about order of requests
  - May be several markov models that have same relative frequency
    - Example of this next

17

Markov Model Example

- Computer network showed packets large (20%) or small (80%)
- 1) sssbssssbssssb

Current Packet	Next packet Small	Next packet Large
Small	0.75	0.25
Large	1	0

2) ssssssssbbssssssssbb

Current Packet	Next packet Small	Next packet Large
Small	0.875	0.125
Large	0.5	0.5

- 3) Or, generate random number between 0 and 1. If less than .8, small else large
  - Next packet is not dependent upon current

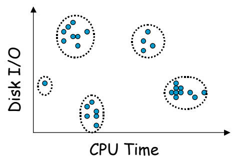
Current Packet	Next packet Small	Next packet Large
Small	0.8	0.2
Large	0.8	0.2

- If performance is affected by order, then need to measure

18

Clustering (1 of 2)

- May have large number of components
  - Cluster such that components within are similar to each other
  - Then, can study one member to represent class
- Ex: 30 jobs with CPU + I/O. Five clusters.



Clustering (2 of 2)

1. Take sample
2. Select parameters
3. Transform, if necessary
4. Remove outliers
5. Scale observations
6. Select distance metric
7. Perform clustering
8. Interpret
9. Change and repeat 3-7
10. Select representative components

(Each step, next)

Clustering: Sampling

- Usually too many components to do clustering analysis
  - That's why we are doing clustering in the first place!
- Select small subset
  - If careful, will show similar behavior to the rest
- May choose randomly
  - However, if are interested in a specific aspect, may choose to cluster only those
    - Ex: if interested in a disk, only do clustering analysis on components with high I/O

Clustering: Parameter Selection

- Many components have a large number of parameters (resource demands)
  - Some important, some not
  - Remove the ones that do not matter
- Two key criteria: impact on perf & variance
  - If have no impact, omit Ex: Lines of output
  - If have little variance, omit. Ex: Pocesses created
- Method: redo clustering with 1 less parameter. Count fraction that change cluster membership. If not many change, remove parameter.

Clustering: Transformation

- If distribution is skewed, may want to transform the measure of the parameter
- Ex: one study measured CPU time
  - Two programs taking 1 and 2 seconds are as different as two programs taking 10 and 20 milliseconds

  → Take ratio of CPU time and not difference

(More in Chapter 15)

Clustering: Outliers

- Data points with extreme parameter values
- Can significantly affect max or min (or mean or variance)
- For normalization (scaling, next) their inclusion/exclusion may significantly affect outcome
- Only exclude if do not consume significant portion of resources
  - Ex: extremely high RTT flows, exclude
  - Ex: extremely long (heavy tail) flow, include

Clustering: Data Scaling (1 of 3)

- Final results depend upon relative ranges
  - Typically scale so relative ranges equal
- Normalize to Zero Mean and Unit Variance
  - Mean $\bar{x}_k$, stddev s_k of the k^{th} parameter

$$x'_{ik} = \frac{x_{ik} - \bar{x}_k}{s_k}$$

- Do this for each of the k paramters

25

Clustering: Data Scaling (2 of 3)

- Weights $x'_{ik} = w_k x_{ik}$
  - Assign based on relative importance
- Range Normalization
  - Change from $[x_{min,k}, x_{max,k}]$ to [0,1]

$$x'_{ik} = \frac{x_{ik} - x_{min,k}}{x_{max,k} - x_{min,k}}$$

- Ex: x_{i1} {1, 6, 5, 11}
  - 1→0, 11→1, 6→ .5, 4→.4
- Sensitive to outliers (say 11 above was 101)

26

Clustering: Data Scaling (3 of 3)

- Percentile Normalization
  - Scale so 95% of values between 0 and 1

$$x'_{ik} = \frac{x_{ik} - x_{2.5,k}}{x_{97.5,k} - x_{2.5,k}}$$

- Less sensitive to outliers

27

Clustering: Distance Metric (1 of 2)

- Map each component to n-dimensional space and see which are close to each other
- Euclidean Distance between two components
  - $\{x_{i1}, x_{i2}, \ldots x_{in}\}$ and $\{x_{j1}, x_{j2}, \ldots, x_{jn}\}$

$$d = \left\{ \sum_{k=1}^{n} (x_{ik} - x_{jk})^2 \right\}^{0.5}$$

- Weighted Euclidean Distance
  - Assign weights a_k for n parameters
  - Used if values not scaled or if significantly different in importance

$$d = \sum_{k=1}^{n} \left\{ a_k (x_{ik} - x_{jk})^2 \right\}^{0.5}$$

28

Clustering: Distance Metric (2 of 2)

- Chi-Square Distance:
  - Used in distribution fitting
  - Need to use normalized or relative sizes influence distance

$$d = \sum_{k=1}^{n} \left\{ \frac{(x_{ik} - x_{jk})^2}{x_{ik}} \right\}$$

- Overall, Euclidean distance is most commonly used

29

Clustering Methodology

1. Take sample
2. Select parameters
3. Transform, if necessary
4. Remove outliers
5. Scale observations
6. Select distance metric
7. Perform clustering
8. Interpret
9. Change and repeat 3-7
10. Select representative components

30

Clustering: Clustering Techniques

- Partition into groups s.t. members are as similar as possible and other groups as dissimilar as possible
  - Minimize intra-group variance or
  - Maximize inter-group variance
- Two classes:
  - Non-Hierarchical – start with k clusters, move components around until intra-group variance is minimized
  - Hierarchical
    - Start with n clusters, combine until k
      - Ex: minimum spanning tree
    - Start with 1 cluster, divide until k

Clustering Techniques: Minimum Spanning Tree

1. Start with $k = n$ clusters.
2. Find the centroid of the ith cluster, $i = 1, 2, \ldots, k$.
3. Compute the intercluster distance matrix.
4. Merge the the nearest clusters.
5. Repeat steps 2 through 4 until all components are part of one cluster.

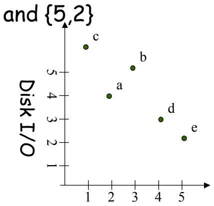
Minimum Spanning Tree Example (1 of 5)

- Workload with 5 components (programs), 2 parameters (CPU/IO).
  - Measure CPU and I/O for each 5 programs

Program	CPU Time	Disk I/O
A	2	4
B	3	5
C	1	6
D	4	3
E	5	2

Minimum Spanning Tree Example (2 of 5)

- Step 1) Consider 5 cluster with i^{th} cluster having only i^{th} program
- Step 2) The centroids are {2,4}, {3,5}, {1,6}, {4,3} and {5,2}

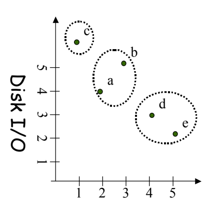


Minimum Spanning Tree Example (3 of 5)

- Step 3) Euclidean distance:

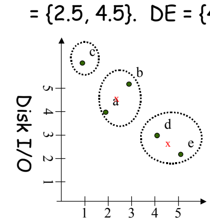
Program	A	B	C	D	E
A	0	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{13}$
B		0	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{13}$
C			0	$\sqrt{18}$	$\sqrt{32}$
D				0	$\sqrt{2}$
E					0

Step 4) Minimum → merge



Minimum Spanning Tree Example (4 of 5)

- The centroid of AB is {(2+3)/2, (4+5)/2} = {2.5, 4.5}. DE = {4.5, 2.5}



Program	AB	C	DE
AB	0	$\sqrt{4.5}$	$\sqrt{10.25}$
C		0	$\sqrt{24.4}$
DE			0

Minimum → merge

Minimum Spanning Tree Example (5 of 5)

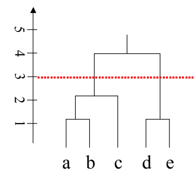
- Centroid ABC {(2+3+1)/3, (4+5+6)/3}
  - = {2,5}

Program	Program: ABC	DE
ABC	0	$\sqrt{12.5}$
DE		0

Minimum
→Merge
→Stop

Representing Clustering

- Spanning tree called dendrogram
  - Each branch is cluster, height where merges



Can obtain clusters
for any allowable distance
Ex: at 3, get abc and de

Interpreting Clusters

- Clusters will small populations may be discarded
  - If use few resources
  - If cluster with 1 component uses 50% of resources, cannot discard
- Name clusters, often by resource demands
  - CPU bound, I/O bound
- Select 1+ components from each cluster as a test workload
  - Can make proportional to cluster size, total resource demands or other