Performance Model Estimation and Tracking using a Kalman Filter

Murray Woodside[1], Tao Zheng[1], Marin Litoiu[2]
[1]Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
[2]IBM Center for Advanced Studies, Toronto
{cmw | zhengtao}@sce.carleton.ca, marin@ca.ibm.com
Sigmetrics 2006/Performance 2006, St. Malo, June 2006.

Part I: What it is, How it Works

Marin Litoiu

- Challenges met by the estimating filter
- The filter concept and history
- An example of its use in an autonomic system

later:

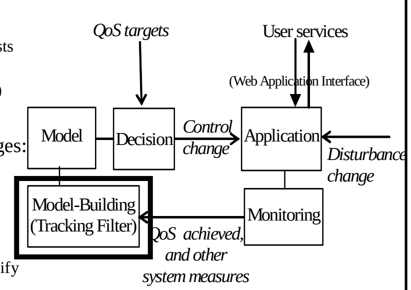
- Part II: Using the Filter for Performance Models
- Part III: Tracking Effectiveness

Challenges Old and New

- the old challenge: to estimate parameters in order to calibrate models
  - our usual approach is to directly monitor the quantity that is the parameter, e.g. CPU time of an operation:
    - intrusive, expensive, time-consuming
- the new challenge: to track parameter changes
  - for adaptive control of dynamically changing systems
    - put a model in the loop
  - measure the running system
    - only at interfaces (source code not available)

Tracking for Model-based Control

- "Disturbance" Changes:
  - rate of requests
  - demands and flows (usage)
- Control Changes:
  - replicas
  - processors
  - allocation
  - threads
  - content (modify demands)



Measure at the System and Component Interfaces

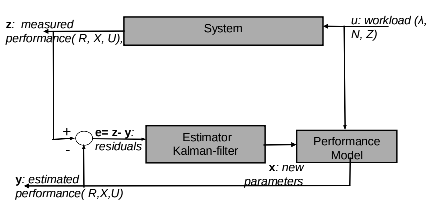
Accessible
- can be made without modifying the system
- can be applied to software components for which source code is unavailable

- measure:
  - event rates
  - response times
  - CPU utilizations
- infer: model parameters such as a service times or routing probabilities

Viewpoint

- We assume the model structure is correct (and perhaps some of the parameter values too)
- We estimate to find parameter values which make the model fit the observations
  - not to validate the structure, for instance
- min mean squared error on the observations

Parameter Estimation

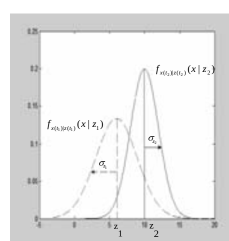


Parameter estimator (Kalman filter): a feedback based system, based on past and current data from the system

Continuously updates the parameters:

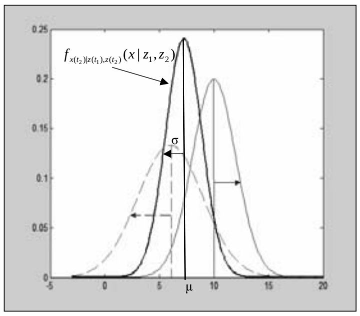
- compares the measured and estimated performance metrics (e)
- adjusts the parameter (state) of the model such that e~0.

A Probabilistic View of the Filter



- Measurement at t_1:
  (z_1, σ_{z_1})
- Best estimate: of x
  $\hat{x}_1 = z_1$
- Measurement at t_2
  (z_2, σ_{z_2})
- Question: Based on the two measurements, what is the best estimate of the x at t_2?

Conditional Density



$\mu = [\sigma^2_{z_2}/(\sigma^2_{z_1}+\sigma^2_{z_2})]z_1 + [\sigma^2_{z_1}/(\sigma^2_{z_1}+\sigma^2_{z_2})]z_2$
$1/\sigma^2 = (1/\sigma_{z_1}{}^2)+(1/\sigma_{z_2}{}^2)$

The best estimate of x at t_2 is

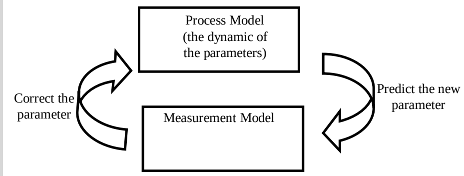
$\hat{x}(t_2)=\mu$

or

$\hat{x}(t_2)=\hat{x}(t_1)+K(t_2)[z_2-\hat{x}(t_1)]$

Predictor Corrector

Predictor -Corrector



The Kalman Filter for Linear Dynamic Systems

- The original filter (1960) was derived to give optimal estimates of time-varying states x_k:
  - Process model: $x_{k+1} = A_k x_k + Bu_k + w_k$
  - Measurement model: $z_{k+1} = H_{k+1} x_{k+1} + v_{k+1}$
  - w_k process noise, with the covariance matrix Q
  - v_k measurement noise, with the covariance matrix R
  - w_k and v_k- white, independent and with a normal distribution
- minimize (in min mean square sense) both the prediction error $(z_{k+1}-H_k\hat{x}_k)$ and the parameter estimation error
  - conditional on:
    - the initial estimates of x_0
    - and P_0...We define P_k = estimated covariance of estimates
    - and the observations z_i over 0 to k

Filter Equations for Linear Systems

- Predict $\mathbf{x}_{k+1}$ and observation $\mathbf{y}_{k+1}$:

  $\hat{\mathbf{x}}^-_{k+1} = \mathbf{A}_k \hat{\mathbf{x}}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{C}_k \mathbf{w}_k$

  $\mathbf{y}_{k+1} = \mathbf{H}_{k+1} \hat{\mathbf{x}}^-_{k+1}$
- Predict the error covariance of $\hat{\mathbf{x}}^-_{k+1}$:

  $\mathbf{P}^-_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$
- Kalman gain **K** :

  $\mathbf{K}_k = \mathbf{P}^-_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}^-_k \mathbf{H}_k^T + \mathbf{R})^{-1}$
- Observe $\mathbf{z}_{k+1}$ and correct the estimate of **x:**

  $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_{k+1} - \mathbf{y}_{k+1})$
- Update the error covariance $\mathbf{P}_k = (I - K_k H)\, \mathbf{P}^-_k$

Kalman Gain

- Minimizes the a posteriori estimate error covariance $E[e_k e_k^T] = \mathbf{P}_k = (I - K_k H)\, \mathbf{P}^-_k$
- Given
  $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_{k+1} - \mathbf{H}_k\hat{\mathbf{x}}_k)$ and
  $\mathbf{K}_k = \mathbf{P}^-_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}^-_k \mathbf{H}_k^T + \mathbf{R})^{-1}$
  - When we have confidence in measurement ($R \to 0$)

  $\mathbf{K}_k = H^{-1} \Rightarrow \hat{\mathbf{x}}_{k+1} = H^{-1}\mathbf{z}_{k+1}$
  - When we have confidence in estimate ($\mathbf{P}^-_k \to 0$)

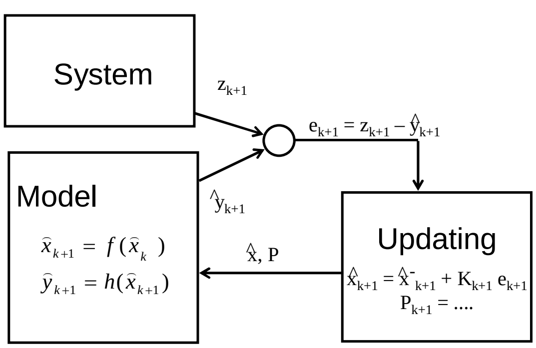
  $\mathbf{K}_k = \mathbf{0} \Rightarrow \hat{\mathbf{x}}_{k+1} = \hat{x}_k$

Convergence

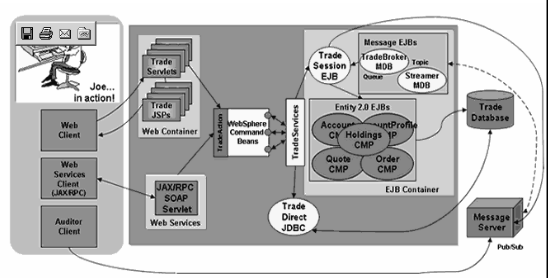
- suppose **x** has size n
- the linear filter converges to a steady state if
  - the state dynamics are controllable (guaranteed if every parameter has a drift term)
  - the state is observable by y. This is satisfied if the observability matrix **O** has rank n

  $\mathbf{O} = [\mathbf{H}^T \;\; \mathbf{A}^T\mathbf{H}^T \;\; (\mathbf{A}^T)^2\mathbf{H}^T \; \ldots\ldots \; (\mathbf{A}^T)^{n-1}\mathbf{H}^T\,]$
  - if $\mathbf{A} = \mathbf{I}$, then the condition is rank($\mathbf{H}$) = n
- this requires at least n *linearly independent* measures, to estimate a state vector **x** of size n.

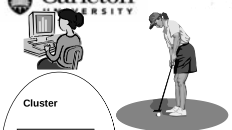
The Extended Kalman Filter (Non-linear Systems)

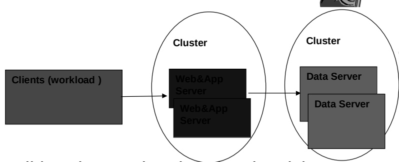


Case Study: Provisioning Trade Application



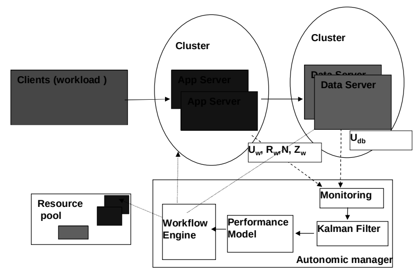
Capacity-on-Demand



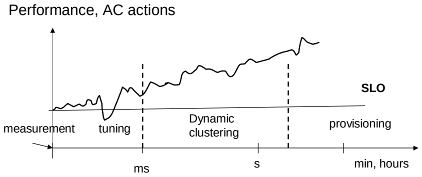


- Traditional capacity planning (static):
  - Alice does capacity planning
- Clustering (dynamic, by human administrators):
  - Alice is system administrator☹
- Autonomic…. Alice plays golf☺

Autonomic Capacity-on-Demand



Different Time Scales for Adapting a System



- There is a time delay between measurement and the end of change execution
  - tuning (ex: change no of threads) can be done in ms
  - provisioning can be done in s, min, hours…
- Without prediction, the adjustments might come too late
  - breaches of SLA, loss of customers…

Monitoring

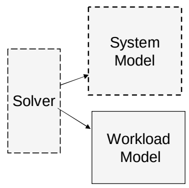
- JMX (Java Management Extension)
  - Implements J2EE javax.management.* interfaces
  - Available with J2EE application servers
  - Provides mean values and variances for J2EE artifacts (servlets, EJBs.. Pools)
- TMTP (Tivoli Monitoring for Transaction Performance)
  - Traces end to end transactions
  - Available for applications implementing ARM
  - Sampling period is too large (hourly…)

Performance Model

a) Workload model
- No of users; Arrival rate; Workload type (or mix)
- Classes of transactions

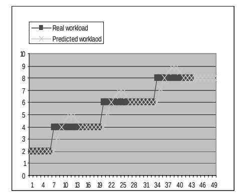
b) System model
- Mimics the system from performance point of view
- analytic models of the system
  - QNMs and LQMs

Solver: matches (a) and (b)
- What is the response time, throughput, etc… for a specific workload(100 users)?
- What if I add 2 App servers?
- The Autonomic Manger queries the Solver, not the real system

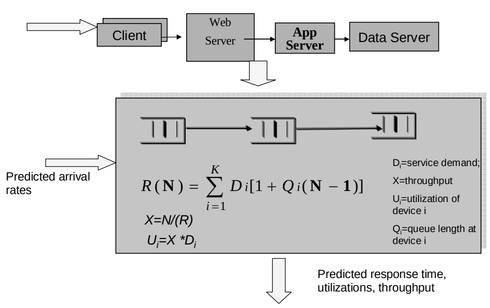


Workload Model

- Closed models: number of users, think time, classes of requests
- Open models: arrival rate, classes of requests
- Measurement based on standard interfaces
- Estimation/prediction based on time series

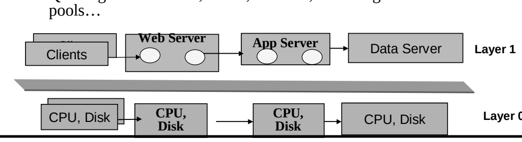


System Model: Queuing Network Model



Layered Queuing Model: Software and H/W

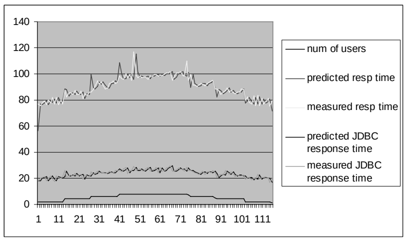
- Layered Queuing Models (LQM) are analytic performance models that
  - Extend Queuing Network Models (QNMs)
  - Model queuing at software components: threading and data connection pools, locks and critical sections
  - Model multiple classes of requests
- LQM Structure
  - Software resource interactions: synchronous, asynchronous, forward call
  - **Demands** at hardware resources for each class of request, one user per class in the system
  - Queuing centers: CPU, DISK, network, threading and data connections pools…



Performance Model Estimation and Tracking using a Kalman Filter

Carleton UNIVERSITY

Model + Estimator : Accuracy



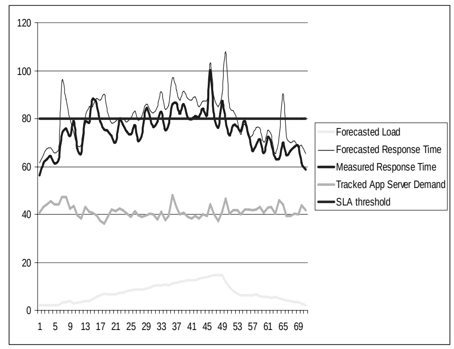
- Measured: servlet response times and CPU utilizations on both tiers, throughput
- Estimated: transaction demands at each tier, no of invocations

25

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Carleton UNIVERSITY

Control Loop Open (i.e., no Provisioning)

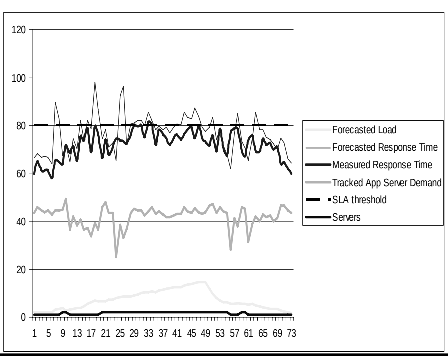


26

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Carleton UNIVERSITY

Control Loop Closed (Provisioning Enabled)



27

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Carleton UNIVERSITY

Conclusions from the Application Study

- The closed loop control successfully controlled the response time to the desired range
  - eliminated the peaks in the graph, that violated the SLA
- Thus, the tracked model was successful in capturing the performance relationships.
- ....next... more details about how models are constructed.

28

Part II: Using the Kalman Filter for Performance Models

Murray Woodside

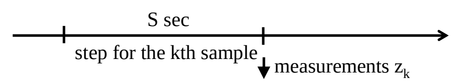
- Filter details for performance models
- Parameter values
- Filter details for Closed Queueing Network (MVA) model
- Estimation effectiveness and parameter tuning
- Issues

later:

- Part III: Tracking Effectiveness

The Filter, Used for Performance

- x = model parameters
  - $x_{k+1} = x_k$: constant parameters (for pure estimation)
  - $x_{k+1} = x_k + w_k$: random drift
  - or $x_{k+1} = A\, x_k + w_k$: autoregressive process for x
- z = vector of measurements
- y = h(x) = the same quantities, as they are predicted by the performance model (*nonlinear*)
- observations are averages over a measurement step time of length S:



Extended Kalman Filters

- For nonlinear dynamics of **x** (not needed here)
- And for nonlinear output function

  $\mathbf{y} = \mathbf{h}(\mathbf{x})$

  In a performance model
  - **x** is the vector of parameters
  - **y** is the vector of predicted measurement values
  - components of **y** match those of the measurement vector **z**
- In the filter gain:
  - replace **A** by $\partial \mathbf{f}(\mathbf{x})/\partial \mathbf{x}$ and
  - replace **H** by $\partial \mathbf{h}(\mathbf{x})/\partial \mathbf{x}$.....
  - evaluated at the predicted estimates

Filter Equations for Performance Models (for $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_k$)

- Prediction of $\mathbf{x}_{k+1}$ is the same as $\mathbf{x}_k$ ($\hat{x}^-_{k+1} = \hat{x}_k$)
- Find $\mathbf{H}_{k+1} = \partial\, \mathbf{h}(\hat{x}_k)/\partial \mathbf{x}$
- Predict the covariance of $\hat{x}^-_{k+1}$:

  $\mathbf{P}^-_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$
- Kalman gain **K** :

  $\mathbf{K}_{k+1} = \mathbf{P}^-_{k+1}\mathbf{H}_{k+1}^T(\mathbf{H}_{k+1}\mathbf{P}^-_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R})^{-1}$
- Correct the state vector:

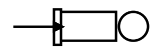
  $\hat{x}_{k+1} = \hat{x}_k + \mathbf{K}_{k+1}(\, \mathbf{z}_{k+1} - \mathbf{h}(\hat{x}_k)\,)$
- Correct the error covariance $\mathbf{P}_{k+1}$:

  $\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}^-_{k+1}$

Iterative Extended Filter (IEKF)

- repeat the update several times, using the new value of $\hat{\mathbf{x}}_{k+1}$ as the starting point for the update, and the same value for **z**.
  - more rapid convergence in the presence of a nonlinear output function, as here.

A Simple Example



- an M/M/1 model, with
  - parameters = $(x(1), x(2))^T$ = (utilization u, service time s)
    - they could equally be: (arrival rate, service time)
  - measurements = $(z(1), z(2))^T$ = (arrival rate f, response time r)
- model is
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_k$
  - $\mathbf{y}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1})$

  and in components of **y**:
  - $\mathbf{y}_{k+1}(1) = \mathbf{h}(1)(\mathbf{x}_{k+1}) = \mathbf{x}_{k+1}(1) / \mathbf{x}_{k+1}(2) \quad = u/s \quad = f$
  - $\mathbf{y}_{k+1}(2) = \mathbf{h}(2)(\mathbf{x}_{k+1}) = \mathbf{x}_{k+1}(2) / [1 - \mathbf{x}_{k+1}(1)] = s/[1 - u] = r$

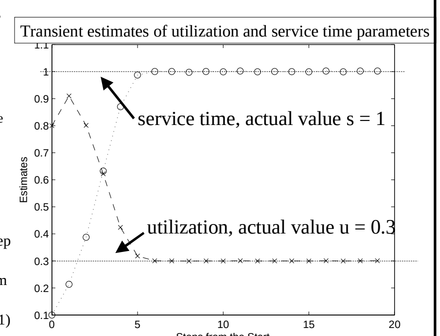
Simple Example (2)

- Linearization of the prediction function:
  - **H**k+1 = $\begin{matrix} 1/s & -u/s^2 \\ s/(1-u)^2 & 1/(1-u) \end{matrix}$

$$= \begin{matrix} 1/\mathbf{x}_{k+1}(2) & -\mathbf{x}_{k+1}(1)/\mathbf{x}_{k+1}(2)^2 \\ \mathbf{x}_{k+1}(2)/[1-\mathbf{x}_{k+1}(1)]^2 & 1/[1-\mathbf{x}_{k+1}(1)] \end{matrix}$$

Simple Example (3): Some Results

- arrival rate 0.3/s
- service time 1 s.
- measure:
  - arrival rate
  - response time
- estimate
  - utilization
  - service time
- measurement step = 100000 s.
- R estimated from simulations
- Q =diag (0.1, 0.1)



Influence of the Filter Parameters Q, R

- Filter *gain matrix*

$$\mathbf{K}_{k+1} = \mathbf{P}^{-}_{k+1}\mathbf{H}_{k+1}^{T}(\mathbf{H}_{k+1}\mathbf{P}^{-}_{k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R})^{-1}$$

- larger **Q** makes **P** larger, and the gain matrix larger
  - intuitively, the filter is “prepared” to see larger changes after each step
  - with **Q** = **0**, **P** converges to **0** (if the filter converges)
  - with **P** = 0 the gain is 0
- larger **R** makes the gain matrix smaller
  - intuitively, the filter has less trust in the measurement value if the error is larger
  - so it responds less to prediction error.
  - even with **R** = 0, the gain is not 0.

Effect of the Estimation Time Step S (1)

- during one time step, the system parameters can drift
  - so, larger S means larger **Q**
- measurements are averages over the time step
  - so, larger S means more accurate averages and smaller R
- to quantify this, consider the drift
  - suppose it is a process of independent increments at some fine time-step, and S contains *k* fine steps of fixed length:
    - $\mathbf{w} = \Sigma_{i=1}^{k} \boldsymbol{\omega}_i$, where $\boldsymbol{\omega}$ has covariance matrix $\boldsymbol{\Theta}$ = diag($\boldsymbol{\theta}$)
  - over one step, drifts are independent
  - then $\mathbf{Q} = k\,\boldsymbol{\Theta}$
  - **Q** is proportional to *k*, i.e. to the step length S.

Effect of the Estimation Time Step S (2)

Effect on **R**:

- **R** represents the covariance matrix of measurement errors
  - the errors reasonably may be assumed independent, so **R** is diagonal, **R** = diag (**v**)
  - where v_i is the variance of errors in z_i
- larger S means more accurate estimates
  - variance ~ 1/(number of samples)

  At a constant rate of sampling:
  - variance ~ 1/S

Effectiveness: Two Questions

1. Estimation: can a KF converge to good estimates from some (incorrect) starting point?
2. Tracking: can it track the parameters when they change?

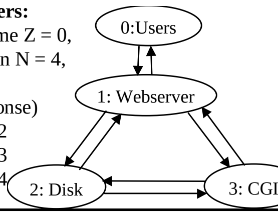
- We shall consider the first question first.
  - aspects to be evaluated:
    - effect of starting estimate
    - speed of acquisition
    - accuracy of estimation
    - sensitivity to **Q** and **R**, and to incorrect values for **Q** and **R**.
- the second question is considered in Part III.

Evaluation on a Closed Queueing Network

For nonlinear filtering, we must evaluate from experience. We will consider an example in detail:

- the system is a known queueing network with constant parameters
- measurement data was generated by simulating the QN

Parameters:
Think Time Z = 0,
Population N = 4,
Demands (sec/response)
D(1) = 2
D(2) = 3
D(3) = 4

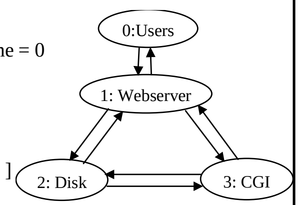


Potential Measurements:
Throughput f
Node delays
T(1)
T(2)
T(3)
Node utilizations
U(1), U(2), U(3)

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

QN: Base Case

- N = 4 users, Z = think time = 0
- **x** is taken to be
  D = D(1), D(2), D(3)
  actual values = [2, 3, 4]
- **z** measured is
  z = [T(1), T(2), T(3), f]
- step length S varies...



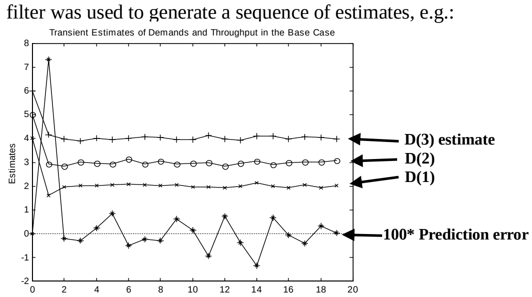
Measurement is over a sampling period of length S

- for S1 = 100000 time units, the variances of elements of **z** were measured as, in order:
  v(S1) = [0.0374, 0.0745, 0.0000737, 0.0109]
- for other (large) values of S, the statistics of averages gives
  v(S) ≈ (S1/S) * **v**(S1))
- set filter parameter **R** = diag(**v**(S))

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Estimation: Filter Transient Response

- initial estimates x_0 were set to [4, 5, 6]
  - compared to actual values [2, 3, 4]
- filter was used to generate a sequence of estimates, e.g.:



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Closed QN Model by MVA: H matrix

- for H we need the derivatives of performance values w.r.to parameters
- for an exact MVA calculation, the MVA equations can be differentiated to get equations for the derivatives
  - like the MVA equations, they are recursive in the population

MVA: Linearization of the Prediction Function

- The exact recursive mean value analysis equations for a separable queueing network are [6], at population N:

  $T(i)^N = (N(i)^{N-1} + 1)\ D(i)$, i = 1,.., n

  $f^N = N / \Sigma_i\ T(i)^N$

  $N(i)^N = f^N\ T(i)^N$, i = 1,..., n

- where:
  - N = the population of jobs or customers in the model,
  - $N(i)^N$ = mean jobs at node i, at population N,
  - $T(i)^N$ = residence time at node 1 per system response, at population N
  - f^N = system throughput at population N,
  - D(i) = demand at node i, per system response.

MVA: Linearization (2)

- For performance, the MVA equations are applied with initial conditions

  $T(i)^1 = D(i)$,

  and are applied for each value of N up to the desired value.
- For *derivatives*, differentiate these equations. Thus for differentiation with respect to D(j), we obtain:

  $\partial T(i)^N / \partial D(j) = \partial / \partial D(j)\ [(N(i)^{N-1} + 1)\ D(i)\ ]$

  $= [\partial N(i)^{N-1} / \partial D(j)]\ D(i)$, i = 1,.., n
- use performance values from the MVA, and derivatives from the previous recursions

MVA: Linearization (3)

- In summary, the derivatives are:

$\partial T(i)^N / \partial D(j) = [\partial N(i)^{N-1} / \partial D(j)]\ D(i)$, i = 1,.., n

$\partial f^N / \partial D(j) = -\ [\ N / (\Sigma_i\ T(i)^N\ )^2\ ]\ \Sigma_i\ \partial T(i)^N / \partial D(j)$

$= -\ (1/N)\ (f^N\ )^2\ \ \Sigma_i\ \partial T(i)^N / \partial D(j)$

$\partial N(i)^N / \partial D(j) = T(i)^N\ \partial f^N / \partial D(j) + f^N\ \partial T(i)^N / \partial D(j)$,

  - with initial conditions $\partial T(i)^1 / \partial D(j) = \delta_{ij}$.

- and the derivatives of U(i), are found from

  $U(i) = f(i)\ D(i)$

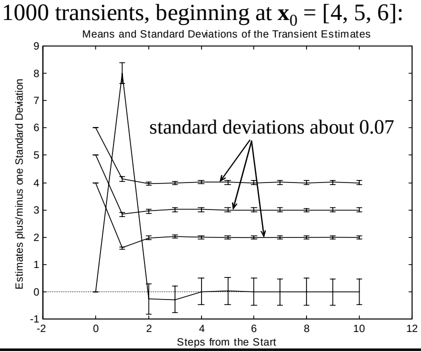
  $\partial U(i)^N / \partial D(j) = D(i)\ \partial f^N / \partial D(j) + f(i)\ \delta_{ij}$.

QN: Drift Matrix Q

- Q(i,i) defines the “assumed” variance of drift of D(i) during one step of length S
- the filter is “prepared” to deal with one-step changes of about √Q(i,i)) in parameter x(i)
- for this study we assumed Q(i,i) = (S/S1)
  - supports tracking change up to about 1 unit of the parameter x(i), per 100000 time units (=S1), for any step size.
  - initial parameter errors were of the order of 1

Estimation Effectiveness (1): Accuracy

- across 1000 transients, beginning at $\mathbf{x}_0 = [4, 5, 6]$:

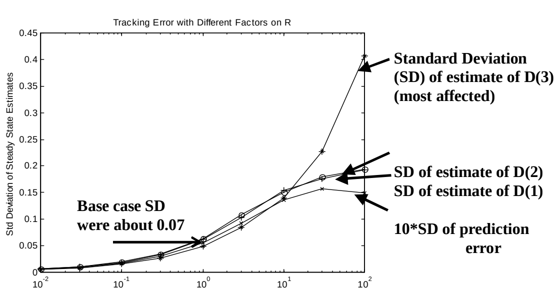


Filter Tuning

- Choice of **Q**, **R** affect the filter gain
- **R** was determined to correspond to the measurement variances (call this $\mathbf{R} = \mathbf{R}^{[*]}$)
  - What if it is not known? How do we set **R**?
  - Does it matter? i.e. are the parameter estimates and the prediction errors sensitive to **R**?
    - Answer = yes
  - Experiment: set $\mathbf{R} = \mathbf{R}^{[*]}$ * Rfactor
    - let Rfactor range from 0.01 to 100
    - find the steady state estimation error *standard deviation* over 1000 steps after step 20

Estimation Effectiveness (2): Tuning R

- smaller R gives more accurate parameter estimation, even when the errors are unchanged



Estimation Effectiveness (3): tuning Q

- made very little difference.
- **Q** governs **P**, which affects the Kalman Gain Matrix K
  - however, the effect seems to be minimal.
  - we conclude that all the values of **Q** are "large enough"
  - there are zero drifts in our system in this case.
- **Q** must not be too small however, this tends to shut off the filter (gains too small).
- Rule of thumb for "large enough":
  - pick a value $\xi(i)$ for each parameter x(i) which is the largest change in x(i) that you would like to track in one step
  - make $Q(i,i) = \xi(i)^2$

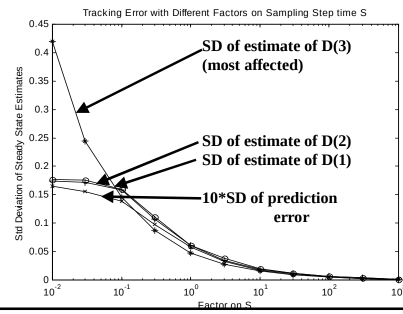
Choice of Step Size S

- We varied S by factors from 0.01 to 1000
- This affects both drift and error (discussed above)
- We applied factors to **Q** and **R** corresponding to the assumptions recorded about the effect of step size:
  - **Q** increases in proportion to S
  - **R** decreases in inverse proportion to S
- The steady state tracking error was again recorded by its standard deviation

53

Estimation Effectiveness (4): step size S

- effect corresponds to the change in **R**



54

The use of P_k as an Error Estimator

- **P** estimates the covariance matrix of the **x** vector
  - meaning, it estimates the variances of estimation error
- $\mathbf{P_k}$ is based on **Q**, **R** and **H**, not on the observed errors in measurements

From:

- $\mathbf{P^-_k} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$
- $\mathbf{K}_k = \mathbf{P^-_k H_k}^T(\mathbf{H_k P^-_k H_k}^T + \mathbf{R})^{-1}$
- $\mathbf{P}_k = (\mathbf{I} - \mathbf{K_k H_k})\mathbf{P}^-_k$

we can write:

- $\mathbf{P^-_k} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$ *(project)*
- $\mathbf{P}_k = \mathbf{P}^-_k - \mathbf{P^-_k H_k}^T(\mathbf{H_k P^-_k H_k}^T + \mathbf{R})^{-1}\,\mathbf{H_k P}^-_k$ *(update)*

55

Effectiveness of P for Estimating Parameter Errors

- If R was correctly estimated, variances (diagonal terms) in P gave good estimates of the actual variances of x:
  - diagonal of P in the base case: [0.0027, 0.0040, 0.0036]
  - measured variances of parameters: [0.0034, 0.0037, 0.0037]

but:

- If R was set too small,
  - measured variances of x were reduced,
  - *variances in P were much smaller*
- if R was set too large,
  - variance of x went up
  - *variances in P were much bigger*

56

Structural Issues

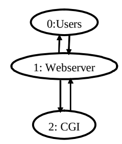
1. Is the correct performance model used in the filter?
   - what happens to the estimates, if not?
   - but, all models are approximations
2. Which measurements to use?
   - in principal, the more the better
   - adding a measurement cannot increase the errors
3. Are the measurements that are available, sufficient?
   - non-convergence with inadequate data
   - the value of additional measurements, for enhanced accuracy

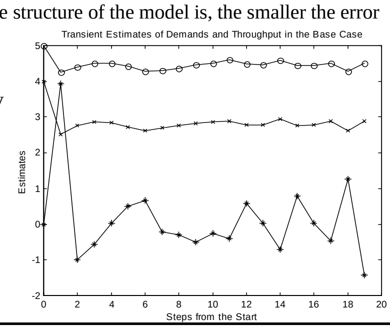
Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Issue (1): Correct Performance Model?

- the filter finds the best fit it can, for the model it is given
- the better the structure of the model is, the smaller the error

Example of a model with only two queues:



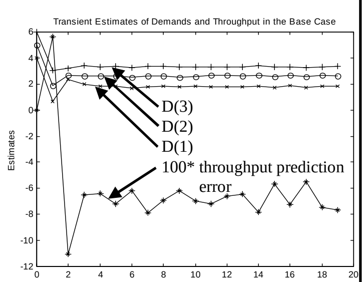


Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Another Incorrect Performance Model

- incorrect value of a parameter which is not estimated
  - An incorrect population (N = 7 in the model, N = 4 in the system)
  - Best fit was not very good, because of internal contradictions

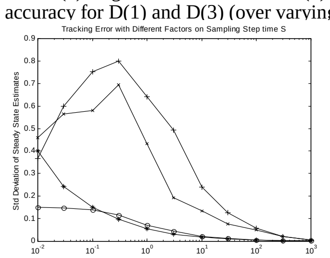
- filter used measures of [T(1), T(2), T(3), f],
- model systematically overestimates congestion
- so, filter under-estimates D
- poor prediction of performance



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Issue (2): Which Measurements?

- Different: z = [T(1), T(2), f] gave slightly larger errors
- Fewer: z = [U(2), f] gave OK estimates of D(2) and f, but poorer accuracy for D(1) and D(3) (over varying S):



- Too few: z = [f] gave arbitrary parameters (that would give good throughput predictions, many solutions)

Issue (3): Enough Measurements?

- From the convergence condition, we know that we must have:

  rank(**H**) = n

- If we have m measurements, H is m by n and we must have:
  - $m \geq n$
  - linearly independent measures. E.g., since in our example

    ResponseTime = *T*(1) + *T*(2) + *T*(3)

    then *ResponseTime* is not linearly independent of the others
  - another example: since *f* = *N/R* and *N* is assumed known, is *f* linearly independent of *R* or of *T*(1)... *T*(3)?

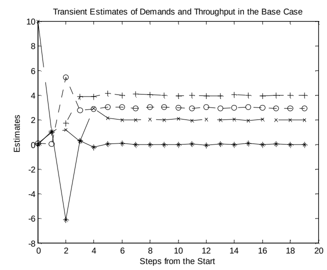
Potential Problem: Bottlenecked System

A bottlenecked model:

- we expect low sensitivity (small elements of **H**) for parameters of non-bottleneck elements. However....

Experiment:
• same system
• initial model was heavily bottlenecked ($\mathbf{x}_1$ = [10, 0.1, 0.1])

• filter converged, but more slowly
• model has sufficient sensitivity to non-bottleneck parameters



Estimating the Population: N

- N enters the MVA equations as an integer, so the linearization to find the H matrix is awkward.
- Our solution: utilize one of the MVA approximations in which N enters as a factor only
  - here, we experimented with the Schweitzer approximation
  - we get a set of simultaneous equations for the derivatives
  - use them as auxiliary equations, only to get the derivatives
  - thus, solve them using the exact solution values for the performance values that also appear as coefficients

Sensitivity of Measures w.r. to N

- Schweitzer approximation:

  $T(i)^N \approx [N(i)^N (1 - (1/N)) + 1] D(i)$,
- This can be differentiated with respect to N to give:

  $\partial T(i)^N / \partial N \approx [\partial N(i)^N / \partial N (1-(1/N)) + N(i)^N (1/N^2)] D(i)$

- evaluation of the derivative uses T(i), N(i) etc from the exact MVA. We also use (found by differentiating the exact MVA equations):

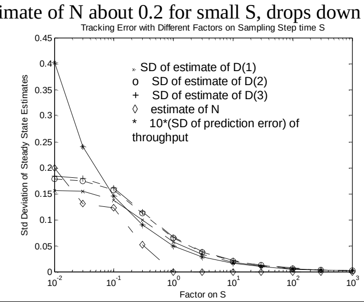
  $\partial f^N / \partial N = 1 / \Sigma_i T(i)^N - (1/N) (f^N)^2 \Sigma_i (\partial T(i)^N / \partial N)$

  $\partial N(i)^N / \partial N = \partial f^N / \partial N \; D(i)$, $i = 1, \ldots, n$
- Three simultaneous nonlinear equations, solved by a fixed-point iteration starting from:

  $\partial N(i)^N / \partial N = 1/K$ (which corresponds to N(i) = N/K for K nodes)

Effectiveness to Estimate N

- Worked well... converged and
  - accuracy comparable to other parameters
  - SD of the estimate of N about 0.2 for small S, drops down to near 0.



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Part III Estimation Effectiveness for Tracking Time-varying Parameters

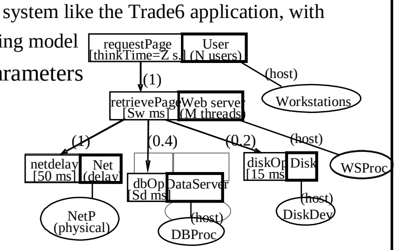
Tao Zheng

- effectiveness on deterministic parameter changes
- effectiveness on random parameter changes
- effectiveness for controlling resource provisioning

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

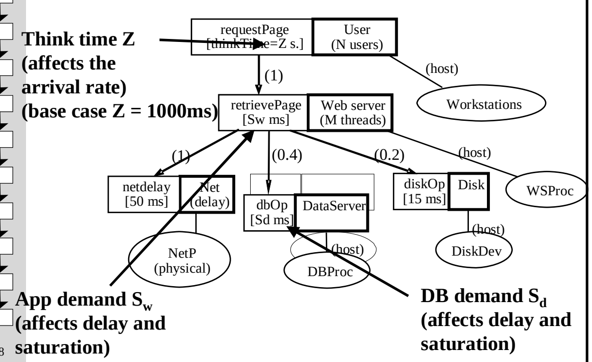
Effectiveness of the Filter for Tracking

- Previous discussion looked at convergence to an unchanging set of parameters
- Now consider a system like the Trade6 application, with
- a layered queueing model
- time-varying parameters



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Model with Time-varying Parameters



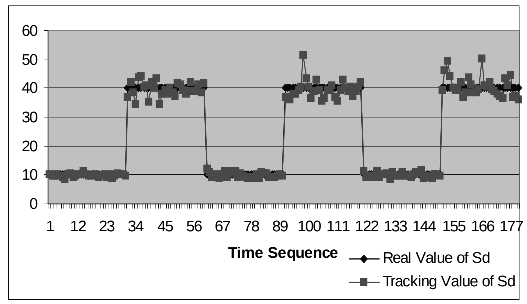
Plan for Evaluation

- Experiments were carried out with different values of:
  - α = the mean rate of change events, whether periodic or random,
  - C = the coefficient of variation of the random values taken by all the parameters xi. parameter values were chosen independently, or according to some pattern.
  - S = the step duration
- To normalize time, a “characteristic step time” S* was defined, long enough to give accurate average response time
  - S* = the value of S which gives confidence intervals of ±5% in response time

    = 15.7 sec in the base case.

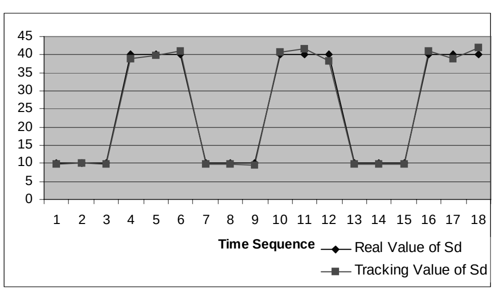
Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Tracking Sudden Deterministic Changes in Database Demand S_d (S = S*)



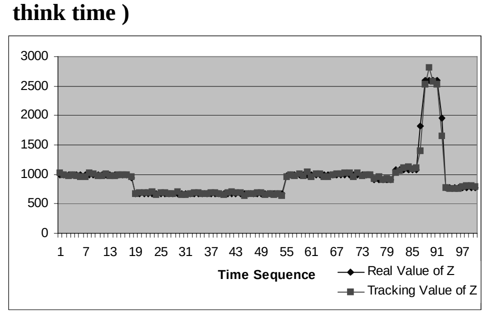
Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Tracking Sudden Deterministic Changes in Database Demand S_d (S = 10S*)



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006

Tracking Random Changes in Workload (due to think time)

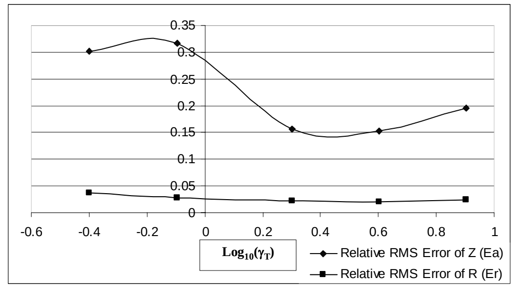


RMS Prediction Error in the User Response Time, as R and Q are Varied

		Q								
		0.01	0.03	0.1	0.3	1	3	10	30	100
	0.01	1.580661	1.580614	1.580513	1.580522	1.580579	1.580594	1.580538	1.580645	1.580615
	0.03	1.580823	1.580642	1.580605	1.58055	1.580504	1.580587	1.580536	1.580633	1.580616
	0.1	1.582551	1.580871	1.580675	1.58066	1.580593	1.580652	1.580635	1.58058	1.58057
	0.3	1.588438	1.582538	1.580831	1.580567	1.580586	1.580542	1.580586	1.580637	1.580469
R	1	1.823365	1.58361	1.582575	1.580895	1.580592	1.580661	1.580553	1.580641	1.580638
	3	2.544668	1.823382	1.588447	1.582593	1.58083	1.580653	1.580478	1.580552	1.580506
	10	4.607373	2.724199	1.823378	1.583604	1.582588	1.580859	1.580682	1.580517	1.580546
	30	6.438646	4.607411	2.544659	1.82338	1.588428	1.58255	1.580838	1.580686	1.580486
	100	8.532533	6.613938	4.607798	2.724203	1.823348	1.583606	1.582568	1.580887	1.58065

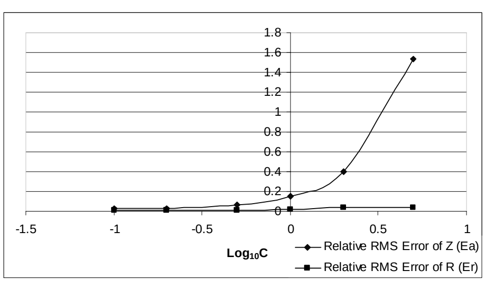
73

Relative RMS Errors in Tracking Random Changes in User Think Time Z, for Different Lengths of the Measurement Interval S (from 0.4S* to 8S*)



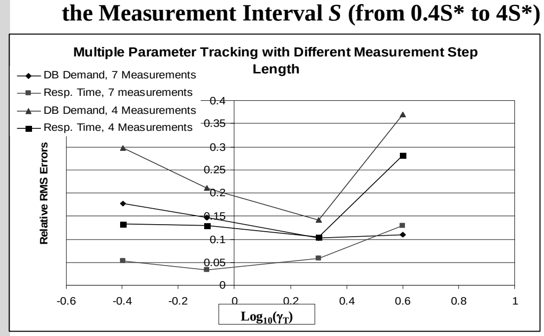
74

Relative RMS Errors in Tracking Random Changes in User Think Time Z, for Different Disturbance Variances C (from 0.1 to 5)



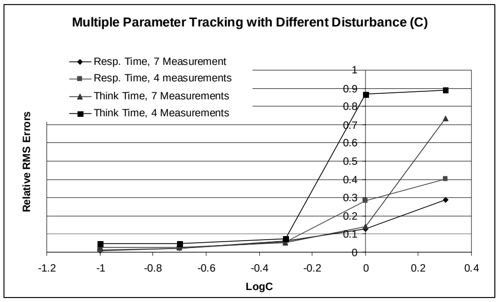
75

Relative RMS Errors in Tracking Random Changes in Multiple Parameters, for Different Lengths of the Measurement Interval S (from 0.4S* to 4S*)



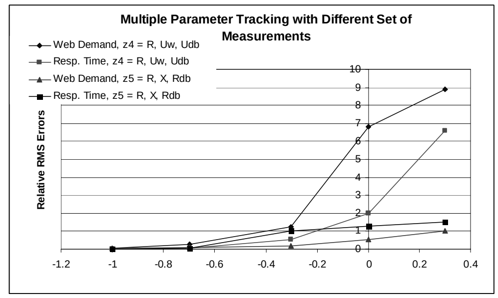
76

Relative RMS Errors in Tracking Random Changes in Multiple Parameters, for Different Disturbance Variances *C* (from 0.1 to 2)



Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Relative RMS Errors in Tracking Random Changes in Multiple Parameters, for Different Set of Measurements



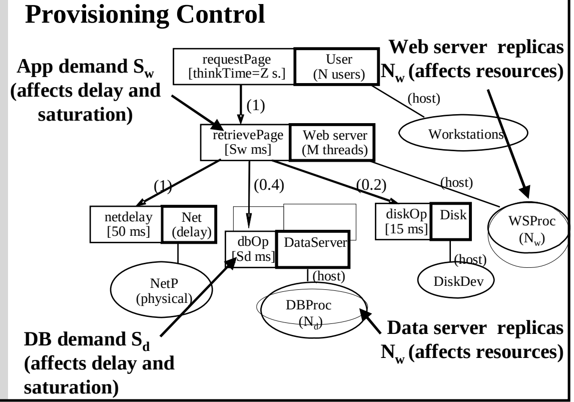
Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Conclusions from Experiments

- Tracking filters work
- The factors affect the tracking quality:
  - Large *S*: small measurement errors but fast change rate
    - Optimal value: balance the accuracy and change rate
  - The ratio of *Q*/*R* rather than *Q*, *R* separately matters
    - Better to overestimate Q or underestimate R
  - The disturbance amplitude
    - smaller *C* is better
- More measurements provide better tracking quality
- Set of response times and throughput seems better than the set of utilizations

Performance Model Estimation and Tracking using a Kalman Filter
© C. M. Woodside, T. Zheng, M. Litoiu, 2006
Carleton UNIVERSITY

Provisioning Control



Servers Provisioning

- The demands (S_a, S_d) of a transaction change over time, taking different combinations (80 in total) of these two sets of values, each combination lasts 10 measurement steps:
  - S_a = {5, 10, 15, 20, 25, 30, 35} ms.
  - S_d = {10, 20, 30, 40, 50, 60, 70} ms
- Web server replicas and data replicas are changed to meet the SLA of user response time
- The SLA is:
  - Mean user response time $R \leq$ 400ms.

Penalty of SLA violation:
  - $Penalty = \sum_{i=1,800} \max(R_{m,i} - 400, 0) / 400$

Provisioning Strategies

- Static Provisioning
  - Fixed number of servers
- Dynamic Provisioning
  - 1. if (R_m > SLA_High)
  - 2.1 Find the minimum number of servers (N_w,N_d) with R_p no more than SLA_High
  - 2. If (R_m <=SLA_Low)
  - 2.1 Find the minimum number of servers (N_w,N_d) with R_p no more than SLA_High
- Perfect Provisioning
  - Always have minimum number of servers to meet SLA

Provisioning Results

- Static Provisioning

DataServer replicas (N_d)	WebServer replicas (N_w)				
	1	2	3	4	5
1	1326.7	639.9	552.9	550.8	548.3
2	956.2	122.1	17.9	17.4	16.4
3	945.7	107.1	0.0	0.0	0.0
4	945.1	106.6	0.0	0.0	0.0
5	941.5	105.5	0.0	0.0	0.0

- Dynamic
  - average number of servers ($N_w + N_d$) = 3.09
  - Penalty = 18.5
- Perfect Provisioning
  - average number of servers ($N_w + N_d$) = 3.08
  - Penalty = 0

Conclusions

- Kalman filters are capable of tracking changing model parameters
- The tuning parameters Q, R must be set to appropriate values for best results (especially R)
- The filter integrates data from many sources, and estimates hidden parameters.
- It can be applied to batch (off-line) data for systems that are not changing, although other approaches such as maximum-likelihood may provide as good or better answers.

Carleton UNIVERSITY

Potential

- tracker can select between model structures
  - by tracking multiple models and choosing the best
- policy manager:
  - parameterize the adaptive changes to be made
  - use heuristic search over these parameters
  - optimization with constraints
- you can insert disturbances or intentional inputs to increase the information flow to the estimators