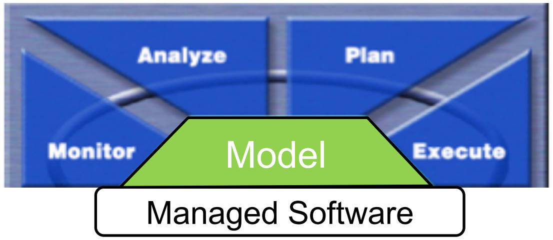
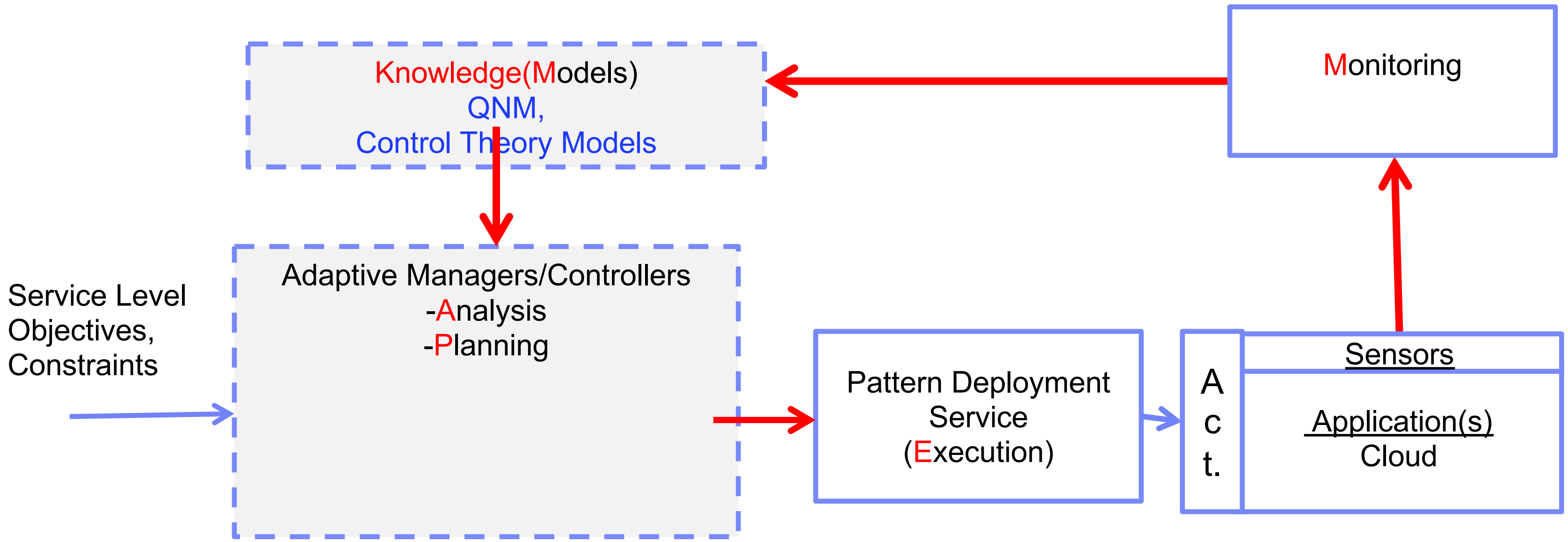
Engineering Model-based Adaptive Software Systems

Marin Litoiu
Adaptive Systems Research Lab
York University
Department of EECS and School of IT
http://ceraslabs.com

Model-based Adaptive Software





MAPE Loop with Control Models

- **Methodology**
  - Have an adaptation goal (set point, objective function, etc…)
  - **Update the knowledge**
    - **Build a model (linear or linearized) of the software system**

$$\begin{cases} x(k+1) & = Ax(k) + Bu(k) \\ y(k) & = Cx(k) + Du(k) \end{cases}$$

    - Constantly update A, B, C, D.
  - Analysis
    - Observability: can I estimate x if I only measure y?
    - Stability: bound inputs-> bound outputs?
    - Controllability: can I reach any x with set of inputs u?
  - **Planning**
    - **Synthesize a controller, e.g**

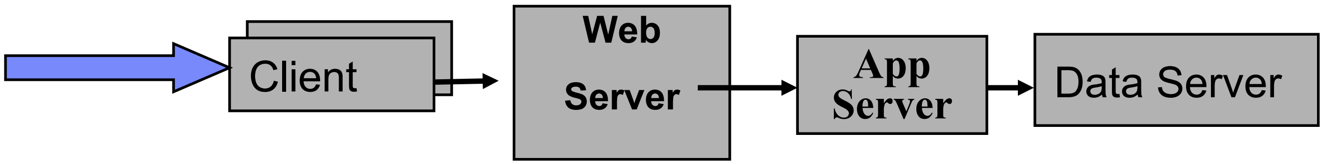
[K, k_r] r=h(goals, A, B,C,D)

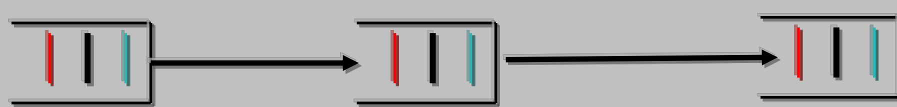
$$K = -Q_u^{-1} B^T P x$$

$$PA + A^T P - PBQ_u^{-1} B^T P + Q_x = 0$$

$$1 = C(A - BK)^{-1} B k_r$$

MAPE Loop with QNMs





Measured # users N, demands D_i, number of servers i etc..

$$R(\mathbf{N}) = \sum_{i=1}^{K} D_i[1 + Q_i(\mathbf{N}-\mathbf{1})]$$

$X=N/(R)$

$U_i=X *D_i$

D_i=service demand;

X=throughput

U_i=utilization of device i

Q_i=queue length at device i

Predicted response time, utilizations, throughput

MAPE Loop with QNM

- **Methodology**
  - Have an adaptation goal (set point, objective function, etc…)
  - **Update the knowledge**
    - **Build a QNM (on-linear)**
      -
    - Constantly update the parameters (d, labmbda,..)
  - Analysis
    - Where is the bottleneck, in what tier?
    - What if the arrival rate increases by x%?
      - How is the response time affected?
  - **Planning**
    - **Synthesize a controller, e.g,**
      - **Find the configuration (#containers, etc..) that achieve the goal while meeting the constraints**

$X=\lambda$

$Ui=X*D_i$

$Ri=Di/(1-Ui)$

$R = \sum_{i=1..K} Ri$

How do we ensure we have accurate models at runtime?

- **Use “estimators” to estimate unmeasurable or uncertain parameters**
- **Use on-line learning and training**

Model Estimation*

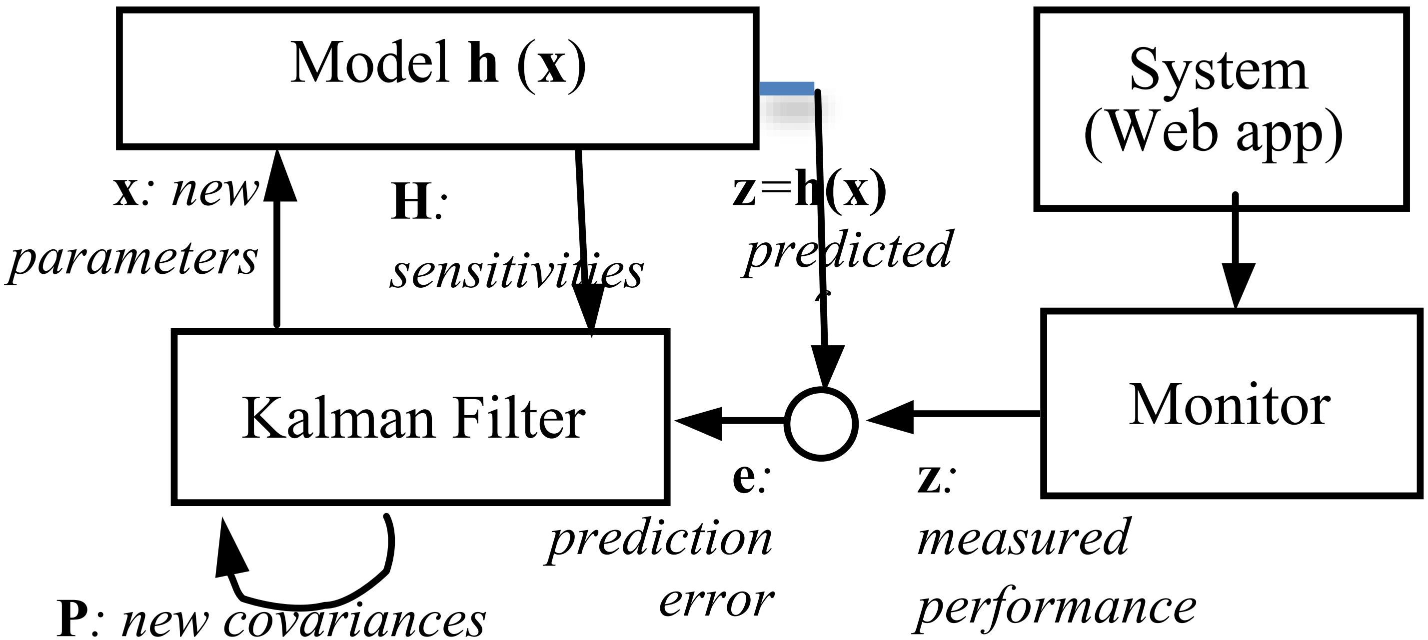
T. Zheng, C. M. Woodside and M. Litoiu, "Performance Model Estimation and Tracking Using Optimal Filters," in *IEEE Transactions on Software Engineering*, vol. 34, no. 3, pp. 391-406, May-June 2008.

- **How do I estimate performance metrics at run time without measuring them?**
  - throughput, response time, cpu or disk service time, arrival rates, number of calls…?
- **Why do we need estimation?**
  - The overhead is too high
  - Measurements have errors
  - To have accurate performance models
- **Additional questions:**
  - Can I track time-varying parameters?
  - Are there limitations?

The Solution (1)

- If $x=[x_1, x_2, ..x_n]$ are the metrics to estimate
- Measure other metrics $y=[y_1....y_m]$
- Have a performance model **y=** ***h*****(x)**
- Find x as a function of y.

The Solution with Kalman Filter(2)



Filter Equations

- Predictor
  - 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 $\hat{\mathbf{x}}^{-}_{k+1}$:

$$\mathbf{P}^{-}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

- Corrector
  - Kalman gain $\mathbf{K}$:

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

  - Observe $\mathbf{z}_{k+1}$ and correct the estimate of $\mathbf{x}$:

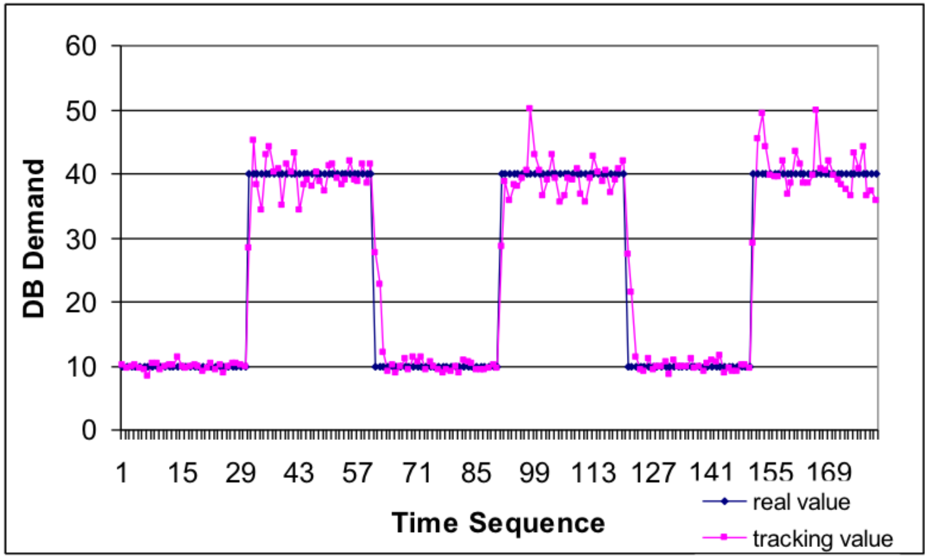
$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k (\mathbf{z}_{k+1} - \mathbf{y}_{k+1})$$

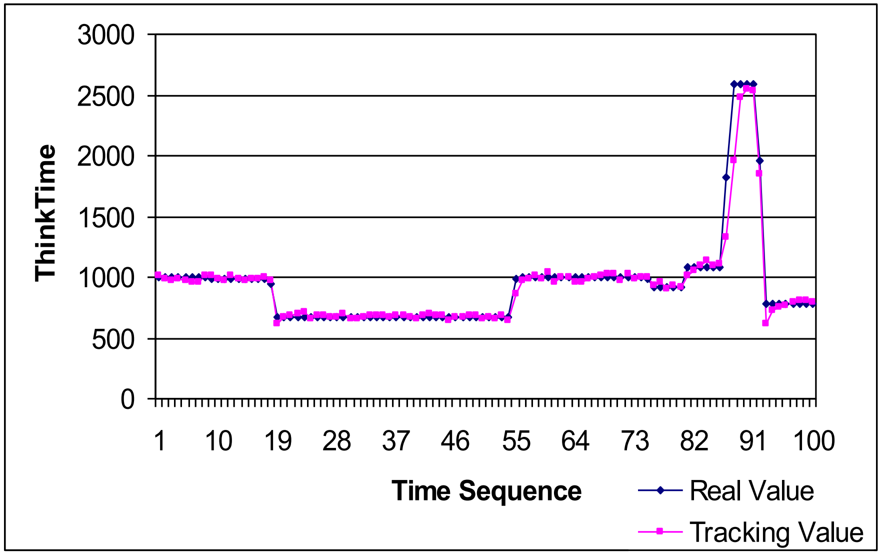
  - Update the error covariance $\mathbf{P}_k = (\mathrm{I} - \mathbf{K}_k \mathrm{H})\, \mathbf{P}^{-}_k$

- **The math**
  - Enables necessary and sufficient conditions
  - Guarantees the convergence
  - Guarantees the optimality
- **We also need**
  - Guidelines for initializing x, P,Q, R
  - A method to find A,B,C,H

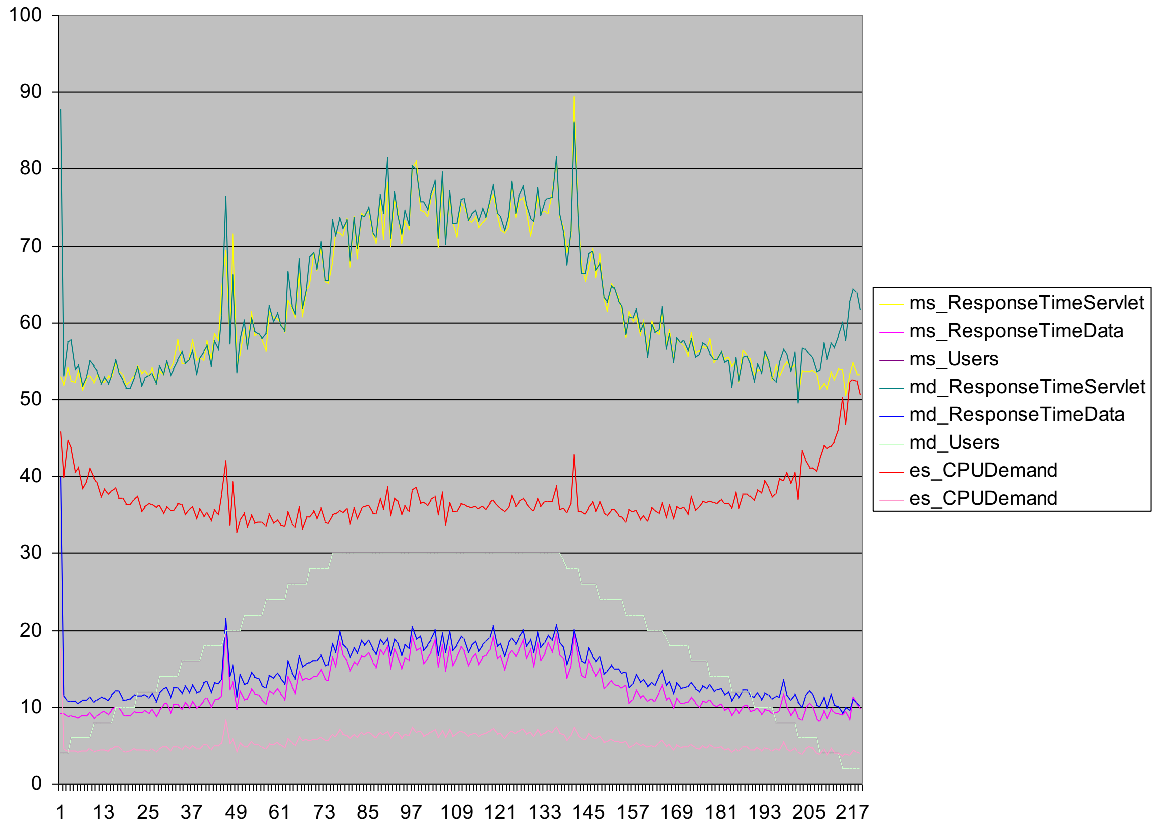
T. Zheng, C. M. Woodside and M. Litoiu, "Performance Model Estimation and Tracking Using Optimal Filters," in *IEEE Transactions on Software Engineering*, vol. 34, no. 3, pp. 391-406, May-June 2008.

Tracking Demands and Think Time in Web Applications



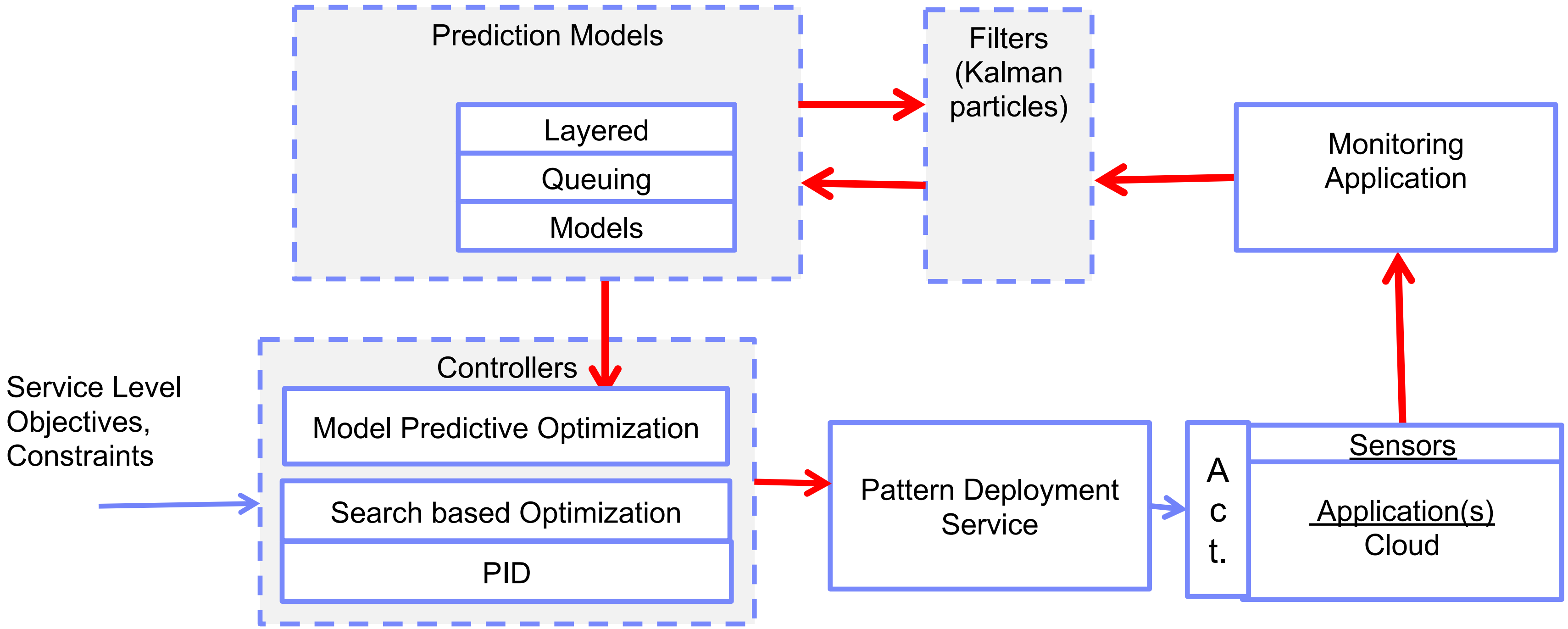


Accuracy of the Performance Model (outputs)**



**M. Woodside, Tao Zheng and M. Litoiu, "Service System Resource Management Based on a Tracked Layered Performance Model," *2006 IEEE International Conference on Autonomic Computing*, 2006, pp. 175-184.

Look Ahead Adaptation

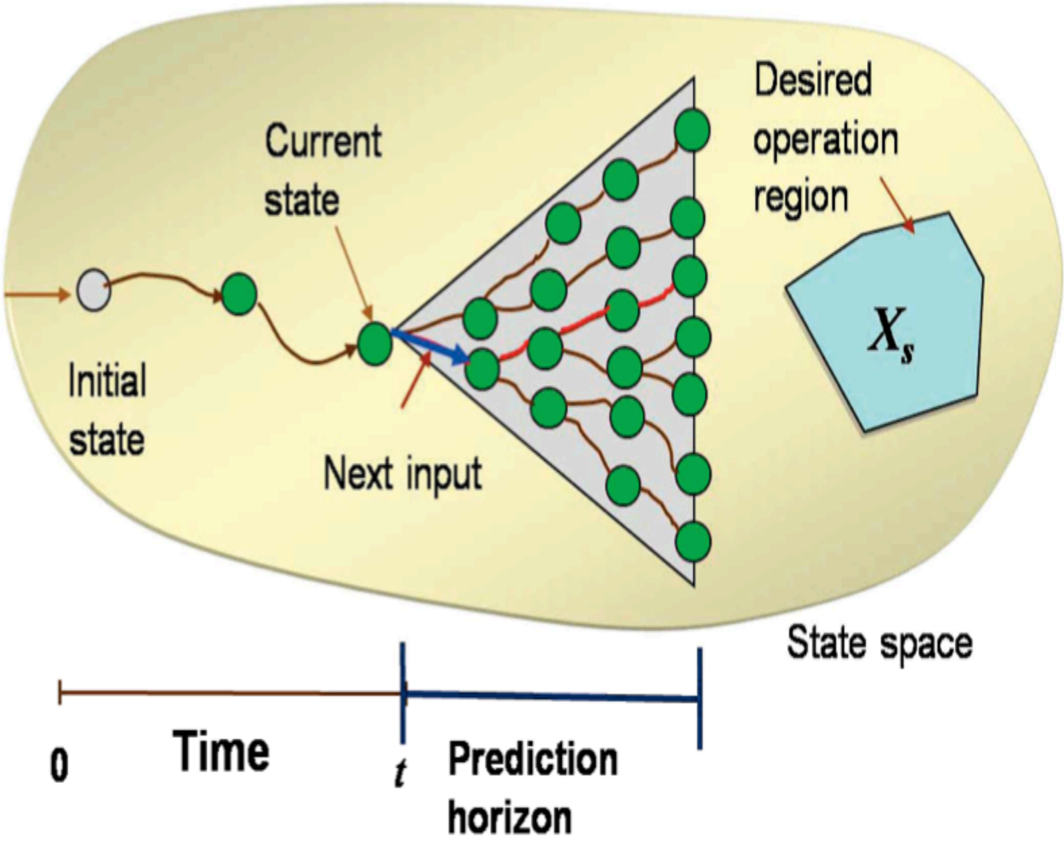


- **Predictive (red arrows): anticipates future load, performance, cost**
  - Uses prediction models, filters and predictive optimization. It is slow and effortful but efficient

Look Ahead Optimization with Thrashing in Cloud

- **The problem:**
  - Given a set of applications in a private cloud, with variable workloads
    - Minimize the cost, performance, trashing
      - By allocating/migrating VMs to physical machines in cloud
- **How it works:**
  - Monitor the current states
  - Predict the workload over a look ahead horizon T
  - Compute the allocation for each step t=1, 2…T
    - For each step, use the model of the system to predict the cost
  - Implement the allocation for step 1
  - Repeat Monitor, Predict, Compute, Implement steps….

Look Ahead Adaptation



From [Bai and Abdelwahed]

Conclusions

Self 1 and Self 2*

System 1

(Self 1)

- **Intuitive**
- **Reactive**
- **Fast**
- **Low resource consumption**
- **Unconscious**
- **Useful but not always right**
- **Ex: 2 x 9=?**

System 2

(Self 2)

- **Analytical**
- **Deliberative**
- **Slow**
- **High resource consumption**
- **Conscious**
- **More often right**
- **Ex: 69 x 37 =?**

Reactive Adaptation ⟷ Model Based Adaptation

*Kahneman, Thinking fast and slow