SmartMig: Risk-Modulated Proactive Data Migration for Maximizing Storage System Utility

Li Yin (U.C. Berkeley)
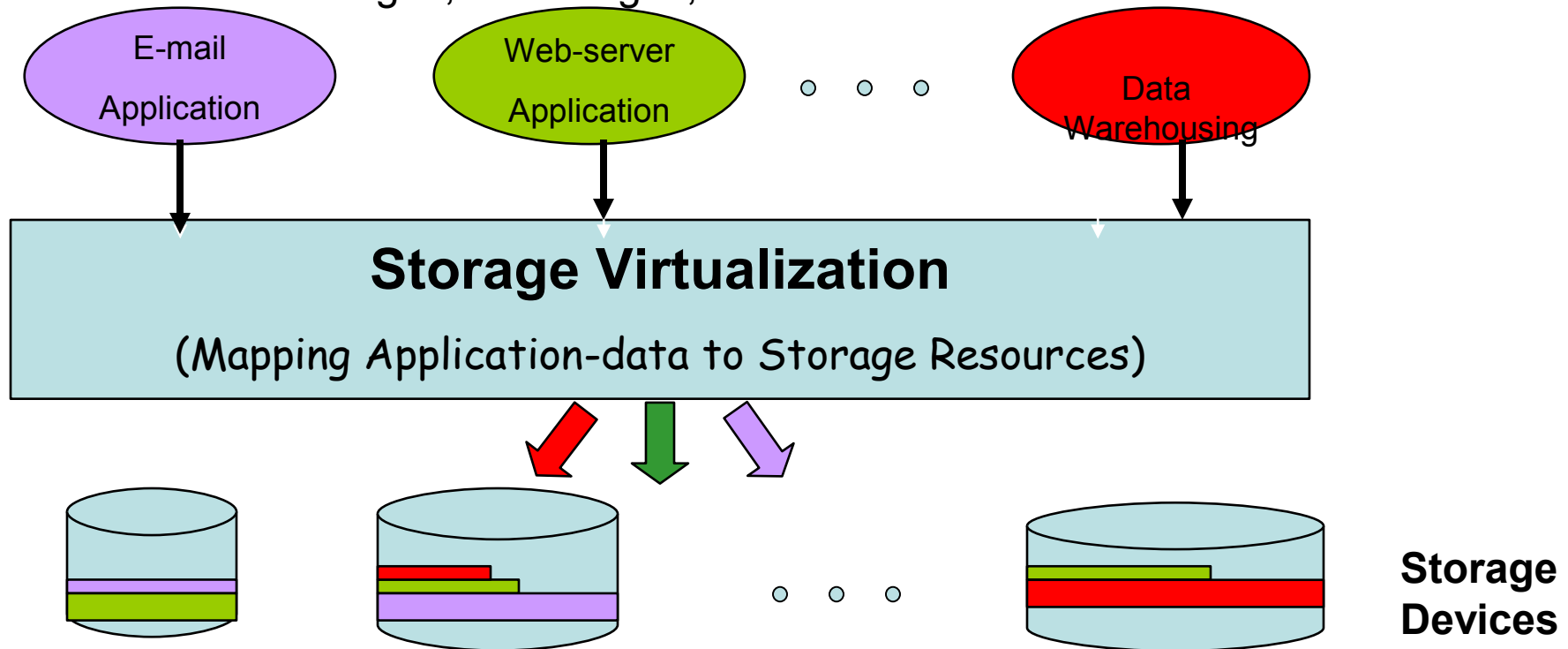
Joint work with:

Sandeep Uttamchandani (IBM Almaden)
Randy H. Katz (U.C.Berkeley)

Need For Corrective Actions

- Increasing trend for storage system consolidation

- Virtualization is the key: determines the application-resource mapping

- Mapping is not static – need for corrective actions
  - Change in application priorities at run-time
  - Workload changes; load surges; hardware failures

E-mail Application

Web-server Application

Data Warehousing

Storage Virtualization

(Mapping Application-data to Storage Resources)

Storage Devices

Corrective actions tune the application-resource mapping

Invoking Data Migration

- Migration is a commonly used corrective action in enterprise storage systems

- Migration invocation parameters:
  - ***What*** to move
  - ***Where*** to move
  - ***How*** to move (migration speed)
  - ***When*** to move

- Existing solutions:
  - Can decide < what, where and how > parameters automatically
    - QoSMig, Aqueduct, Hippodrome
  - <when> is determined by the administrator or using some default policies
    - Example: Invoke the migration when the system load is low

What is Missing?

Unaddressed Problems	When to invoke	Optimization window	Risk Consideration
Existing approaches	When the system is lightly loaded, as a background process	Migration decision is made to optimize the **current** state	No notion of risk: selects migration plans that utilize the system better
Real World Operations	Services are becoming global – systems "never sleep"	The system is optimized with a Look-ahead (optimization) window	Risk is involved: May migrate terabytes of data
Contributions of SmartMig	Migration can be invoked at **any time**, as a foreground process	Account for both the current and **look-ahead** system states	Take the **risk** of migration operation into account

Outline

SmartMig: Utility Based Optimization

- **Goal: Find migration parameters that maximize the system utility for a given optimization window**
- Defining utility functions:
  - Reflect user's satisfaction
  - Defined on a per-workload basis
  - Example Utility functions can be:

$$U(thru, lat) = \begin{cases} 60 * thru & if\ lat < 6ms \\ 10 * thru & if\ lat > 6ms \end{cases}$$

- System utility value:

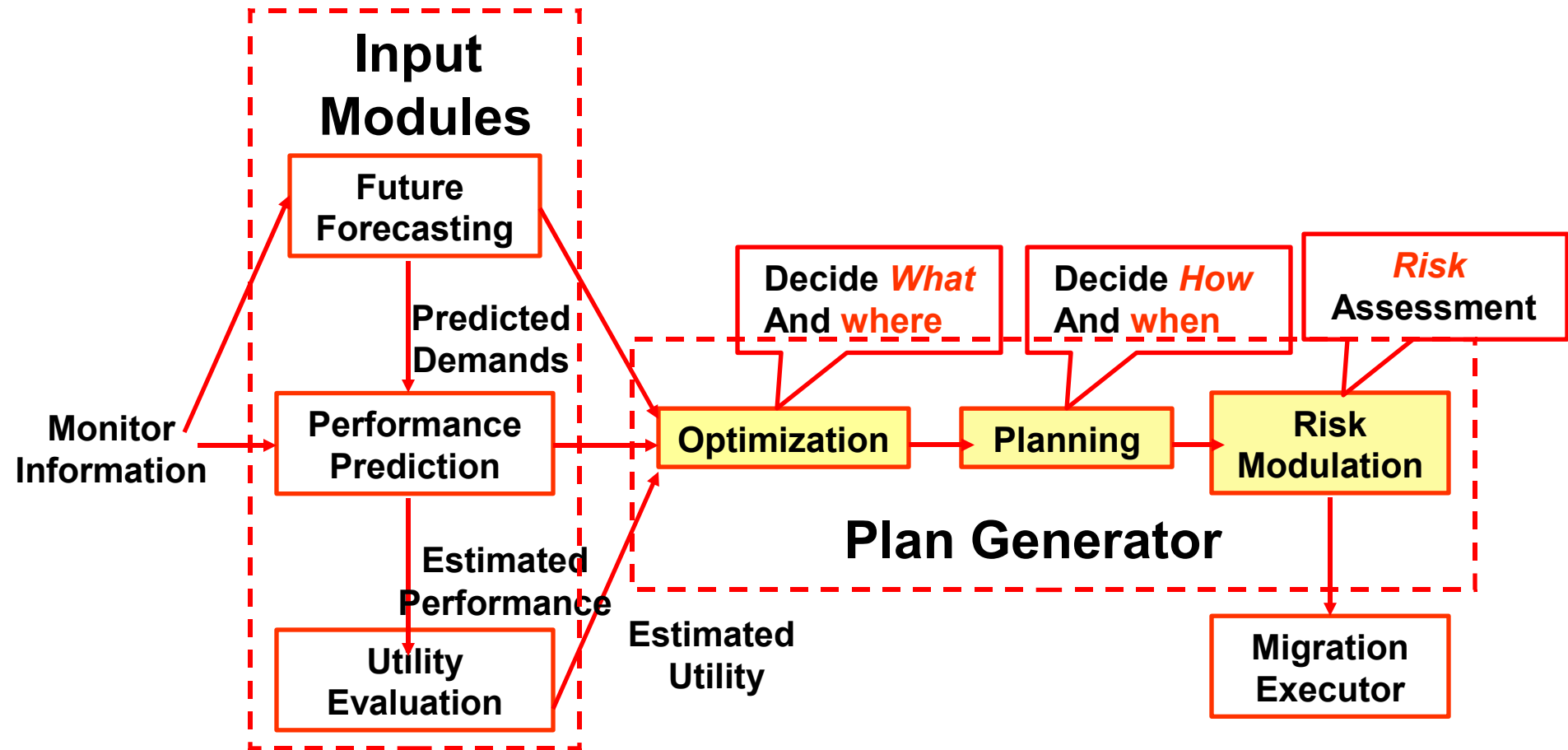$$U_{sys} = \sum_{i=1}^{N} U_i(thru_i, lat_i)$$

- Maximum utility value

$$U_{max} = \sum_{i=1}^{N} U_i(Demand_i, SLO_{lat_i})$$

- System utility loss:

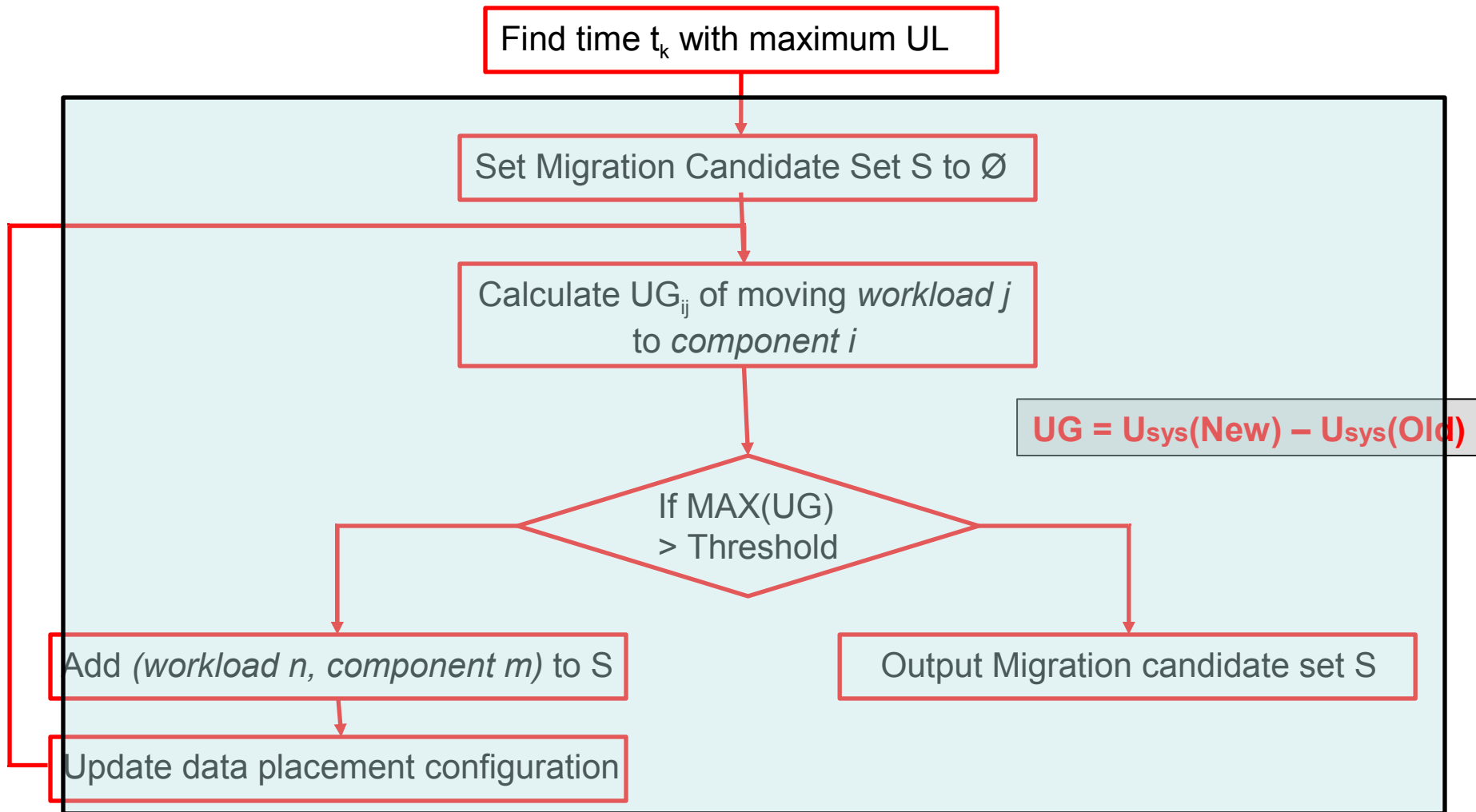$$UL_{sys} = U_{max} - U_{sys}$$

SmartMig Architecture

Outline

Optimization Phase: *What* and *Where*

- **Formulated as a constraint optimization problem**

Find time t_k with maximum UL

Set Migration Candidate Set S to Ø

Calculate UG_{ij} of moving *workload j* to *component i*

UG = U$_{sys}$(New) − U$_{sys}$(Old)

If MAX(UG) > Threshold

Add *(workload n, component m)* to S

Output Migration candidate set S

Update data placement configuration

- **Finding top K <what, where> solutions**
  - **Block the <workload, component> pair leading to minimum UG/Size**

Illustration of Optimization Phase

Migration Candidate Set S= {(2:A→C)} (5:A→B), (3:A→C)}

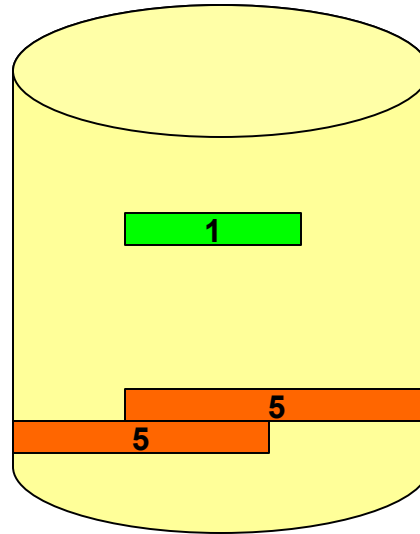UG_{1B} = Utility (W1 on Pool B) – Utility (W1 on Pool A)
UG_{1C} = Utility (W1 on Pool C) – Utility (W1 on Pool A)
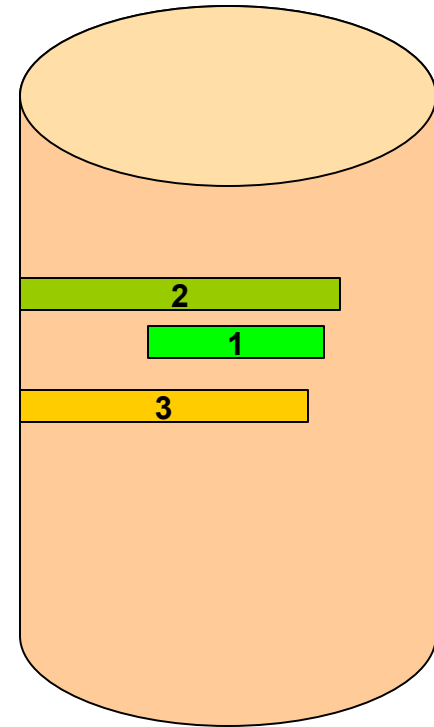**MAX(UG) =** Max(UG_{1B}, UG_{1C}, UG_{2B}, UG_{2C}, …, UG_{6B}, UG_{6C})
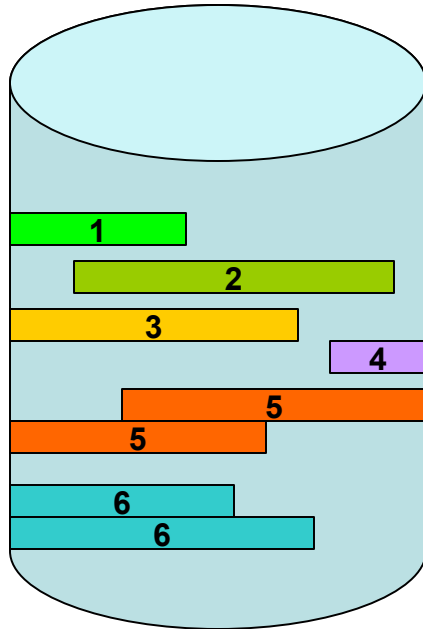
Planning Phase: *How* and *When*

- Migration Speed: optimization problem
  - Greedy approximation – simulated annealing algorithm
- When: choose migration starting time
  - **Goal: find time t with minimum overall system utility loss**
- Overall System Utility Loss: cumulative utility loss across the optimization window
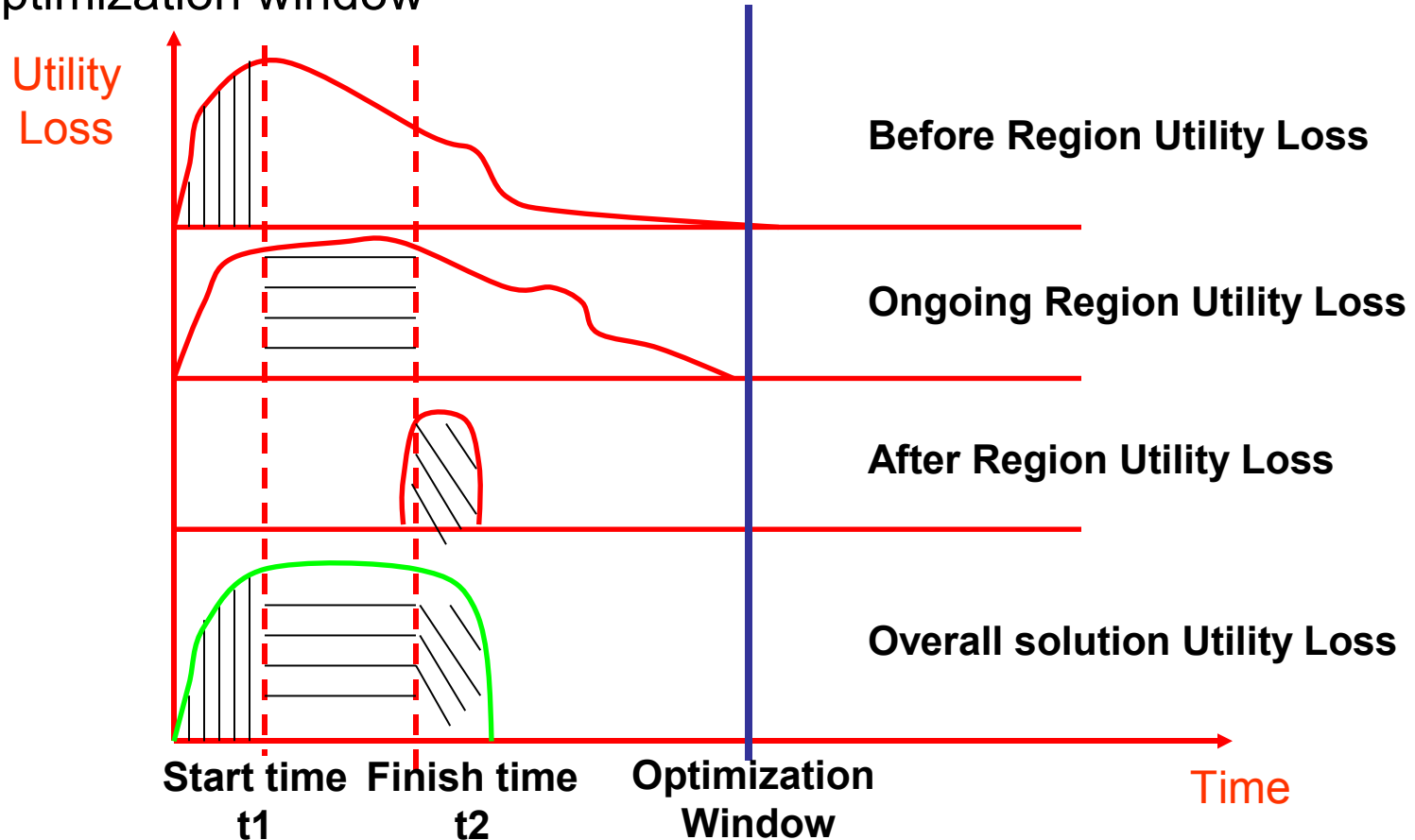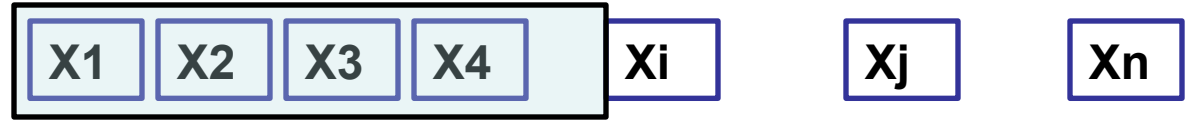
Utility Loss

Before Region Utility Loss

Ongoing Region Utility Loss

After Region Utility Loss

Overall solution Utility Loss

Start time t1 **Finish time t2** **Optimization Window** Time

Illustration of Planning: Choosing Start Time

Utility loss of old data placement at time i

| X1 | X2 | X3 | X4 | | Xi | | Xj | | Xn |

Utility loss of migrating data at time i

| Y1 | Y2 | Y3 | Y4 | | Yi | | Yj | | Yn |

Utility loss of new data placement at time i

| Z1 | Z2 | Z3 | Z4 | | Zi | | Zj | | Zn |

- **For migration option starts at time "i" and ends at time "j"**

Overall UL = Before [0, i) + Ongoing [i, j) + After [j, n]

| X1 | X2 | X3 | X4 | | Yi | | Zj | | Zn |

- **End time "j" is estimated according to the migration speed**

When: the "t" leading to minimum overall utility loss

Risk Modulation Phase

- Account for future uncertainty and migration overhead

- Future uncertainty consideration

$$VaR(95\%Confidence) = -1.65\sigma * \sqrt{T}$$

- Overhead consideration

$$\alpha_{M_k} = \frac{bytes\_moved_{M_k}}{total\_bytes\_on\_source} * Sys\_Util$$

$$RF(M_k) = (-1 + \alpha_{M_k}) * VaR$$

- Scale utility loss for each migration option:

$$UL^*_{M_k} = (1 + RF(M_k)) * UL_{M_k}$$

- If *min(UL*)*<threshold, send for execution. Otherwise, no migration option is returned

Outline

- Motivation/Contributions
- SmartMig
  - Architecture
  - Key Modules
- Evaluation
- Conclusion

Experimental Results

- Three sets of tests
  - Sanity check: parameter's impact on the migration decision
  - Efficiency test: improvement in the system utility
  - Sensitivity test: impact of model errors on the decision

- Test setup
  - Initial data placement: create unbalanced system randomly
  - Workloads features: Gaussian mixture distribution
  - Workload trending
    - Mixture of increasing and decreasing workloads

Sanity Check

- Default settings
  - 20 workloads on 4 components.
  - 14 days optimization window

Option #	Migration Candidates and Targets	Size (GB)	Utility Loss	Start Time (hour)	Scaled Utility Loss
1	(5:0→2) (1:0→2)	19	10408	5	11629
2	(5:0→2) (3:0→3) (4:0→1)	152	22552	4	43715
3	(5:0→2) (3:0→3) (13:0→3)	28	10408	1	12208
4	(5:0→2) (13:0→3) (14:0→2)	118	22552	5	38981
5	(5:0→2) (13:0→3) (17:0→2)	20	10408	4	11694

- Impact of Optimization Window T

#	Migration Candidates and Targets	Size (GB)	Scaled Utility Loss
1	(1:0→ 2)	14	11308
2	(6:0→1)	12	87850
3	(3:0→2) (5:0→2)	17	11501
4	(4:0→2) (5:0→2)	140	42282
5	(5:0→2) (13:0→3)	16	11436

- Impact of Utility Configuration

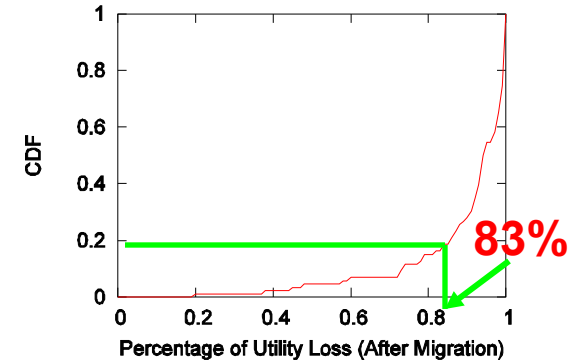| #	Migration Candidates and Targets	Size (GB)	Scaled Utility Loss
1	(17:0→ 2)(3:0→2) (1:0→ 2)	30	2949
2	(17:0→2)(3:0→2)	16	23700
3	(17:0→2) (4:0→2)(5:0→3)	144	15813
4	(17:0→2)(5:0→2)(13:0→ 3)	20	2796
5	(5:0→2) (17:0→2)	9	16200

Efficiency Test and Decision Time

- ## Improvement on system utility
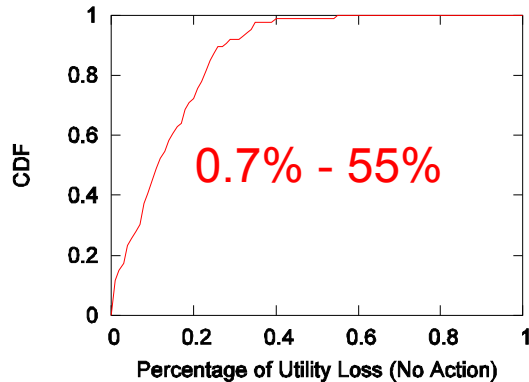
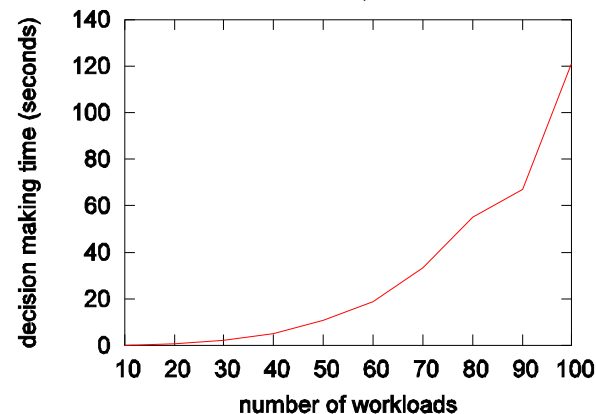(a) CDF of percentage of overall utility loss without migration operation
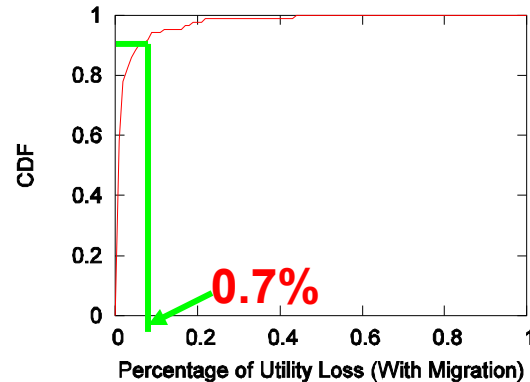
(b) CDF of percentage of overall utility loss with SmartMig

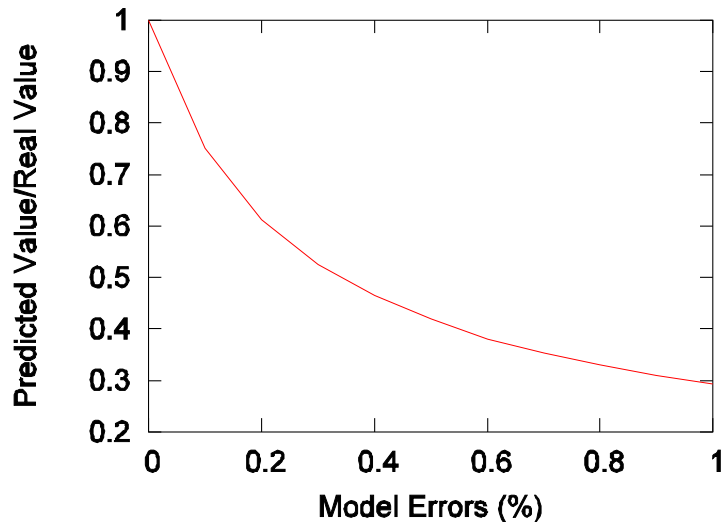(c) CDF of percentage of overall utility loss eliminated by SmartMig

- ## SmartMig Decision Time
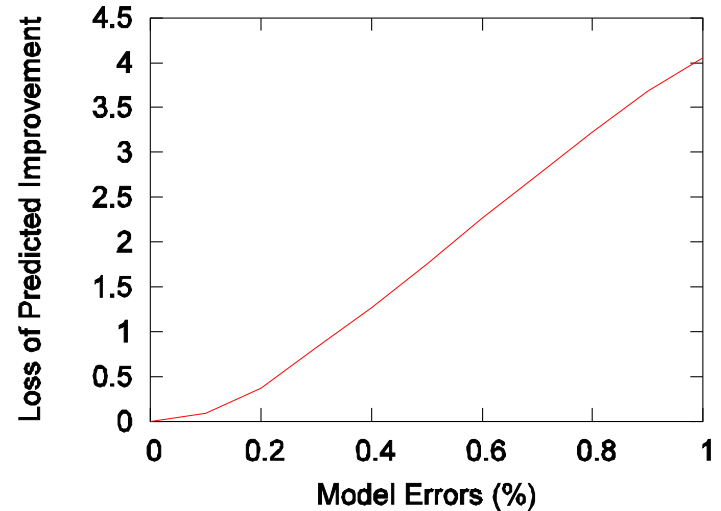 - Linux, Pentium 4, 2.66GHZ CPU, 512MB memory.

Sensitivity Test

- Testing Methodology
  - Synthetic workload models for 'predicted latency'
  - A random error percentage (normal distribution) is applied to generate the 'real latency'

(a) Predicted Utility Loss / Real Utility Loss

(b) (Real Utility Loss – Predicted Utility Loss) / Predict no action utility loss

- Model error should be within 20%
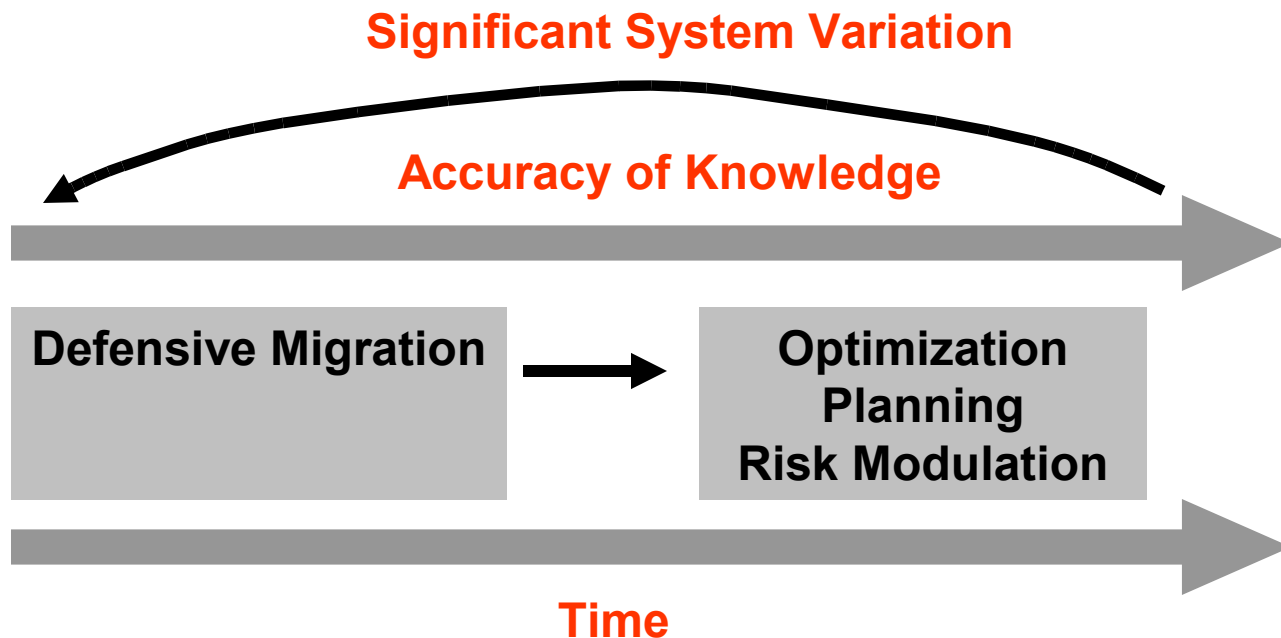
Conclusions

- We proposed SmartMig
  - Decide migration invocation time automatically
  - Account for both the current and future system states
  - Select migration option with minimum risk

- Future work
  - Impact of future prediction errors
  - Real system implementation (GPFS)
  - Improve the decision making time

Questions?

Backup Slides

-

- When future forecasting has large errors
  - Defensive strategy: invoke the migration option with minimum invocation cost and maximum utility gain

Significant System Variation

Accuracy of Knowledge

Defensive Migration

Optimization Planning Risk Modulation

Time

Time Series Prediction

- HP Cello99, Nov/01/99 – Dec/30/99
  - First 41 days as training data
- ARIMA algorithm
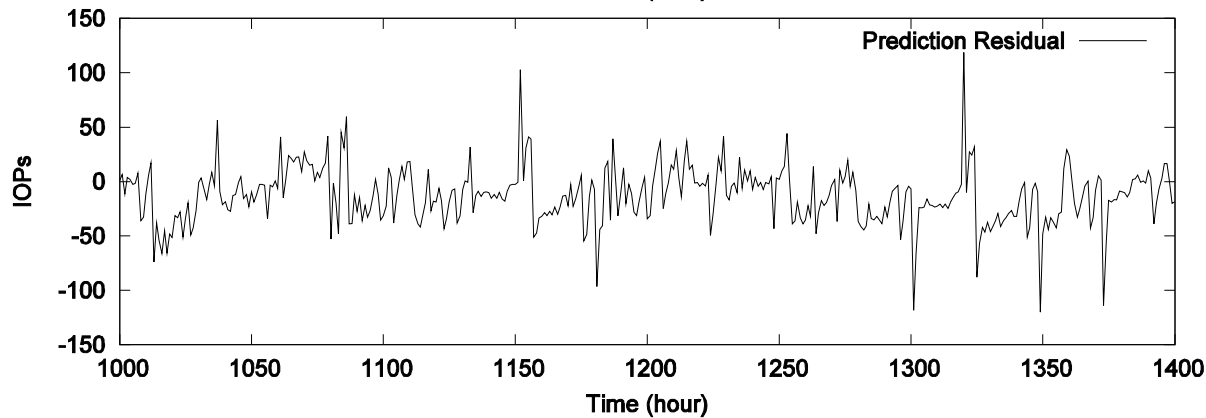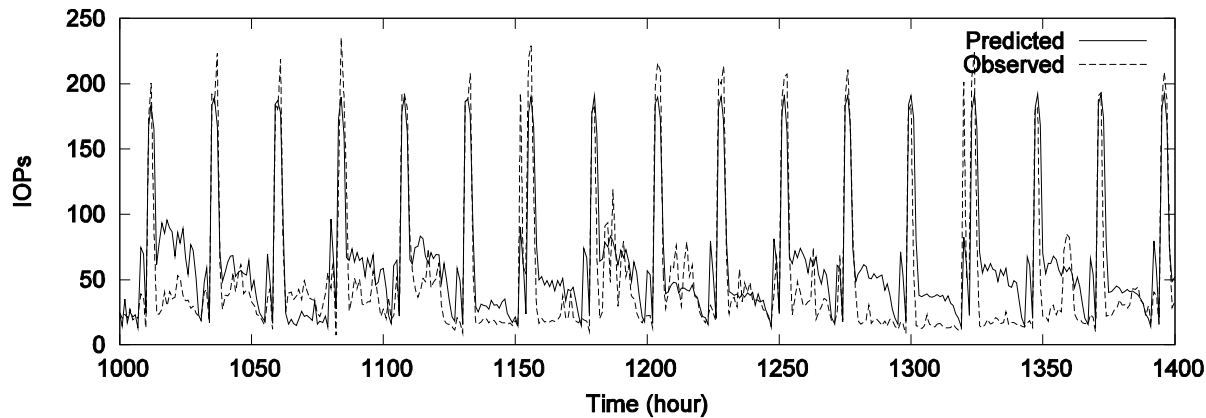
Component Model Examination

- Regression tree based algorithm-GUIDE
- 5 workloads running on GPFS system