應用排隊論分析一個具異質服務處理器的超級電腦系統

王 嘉 宏 博士
[1]福建工程學院信息科學與工程學院
[2]大數據挖掘與應用技術重點實驗室
中國福建省福州市大學新區學園路3號 (郵編: 350118)
個人網頁：http://myweb.ncku.edu.tw/~chwang728/
福建工程學院教師訊息：
http://sise.fjut.edu.cn/s/84/t/566/7f/40/info32576.htm

2016/4/18 東吳大學數學系演講

Outline

- Introduction to Supercomputer CRAY XE6m at NCKU
- A Queueing Model for Two Heterogeneous Processors with a Finite Buffer
- A Computing Approach—K-matrix Based Algorithm
- Numerical Comparisons
- Conclusions and Future Works

什麼是「超級電腦(supercomputer)」？

- 能夠執行一般個人電腦無法處理的大資料量與高速運算的電腦。
- 基本組成元件與個人電腦的概念無太大差異，但規格與效能則強大許多。
- 運算速度達到每秒一兆次(tera flops)以上。

5G

平板

100G

桌機

150G

工作站

420G

伺服器

……

"FLOPS“ floating-point operations per second

Prefix	Symbol	1000^m	10^n
yotta	Y	1000^8	10^{24}
zetta	Z	1000^7	10^{21}
exa	E	1000^6	10^{18}
peta	P	1000^5	10^{15}
tera	T	1000^4	10^{12}
giga	G	1000^3	10^9
mega	M	1000^2	10^6
kilo	k	1000^1	10^3
hecto	h	$1000^{2/3}$	10^2
deca	da	$1000^{1/3}$	10^1

超級電腦速度以每秒浮點運算次數"**FLOPS**“
(floating-point operations per second)來作量度單位

- 一個 MFLOPS (megaFLOPS) 等於每秒100萬 (=10^6) 次的浮點運算
- 一個 GFLOPS (gigaFLOPS) 等於每秒10億 (=10^9) 次的浮點運算
- 一個 TFLOPS (teraFLOPS) 等於每秒1萬億 (=10^{12}) 次的浮點運算
- 一個 PFLOPS (petaFLOPS) 等於每秒1千萬億 (=10^{15}) 次的浮點運算
- 一個 EFLOPS (exascaleFLOPS) 等於每秒100億億 (=10^{18}) 次的浮點運算

Introduction to Supercomputers

超級電腦速度以每秒浮點運算次數"**FLOPS**"
(floating-point operations per second)來作量度單位

1T-FLOPS
每秒做1兆次以上
有小數點數字
加/減/乘計算

1G-FLOPS

Smart phone

10G-FLOPS

Notebook

100G-FLOPS

Workstation

500G-FLOPS

Server

4T-FLOPS
=4,000G-FLOPS

Supercomputer

- 一個 MFLOPS (megaFLOPS) 等於每秒100萬 (=10^6) 次的浮點運算
- 一個 GFLOPS (gigaFLOPS) 等於每秒10億 (=10^9) 次的浮點運算
- 一個 TFLOPS (teraFLOPS) 等於每秒1萬億 (=10^{12}) 次的浮點運算
- 一個 PFLOPS (petaFLOPS) 等於每秒1千萬億 (=10^{15}) 次的浮點運算
- 一個 EFLOPS (exascaleFLOPS) 等於每秒100億億 (=10^{18}) 次的浮點運算

VP2600(1988)
5G FLOPS

Smartphone*
(2011)
4.9G FLOPS

*ARROWS X LTE made by Fujitsu

x 2,000,000

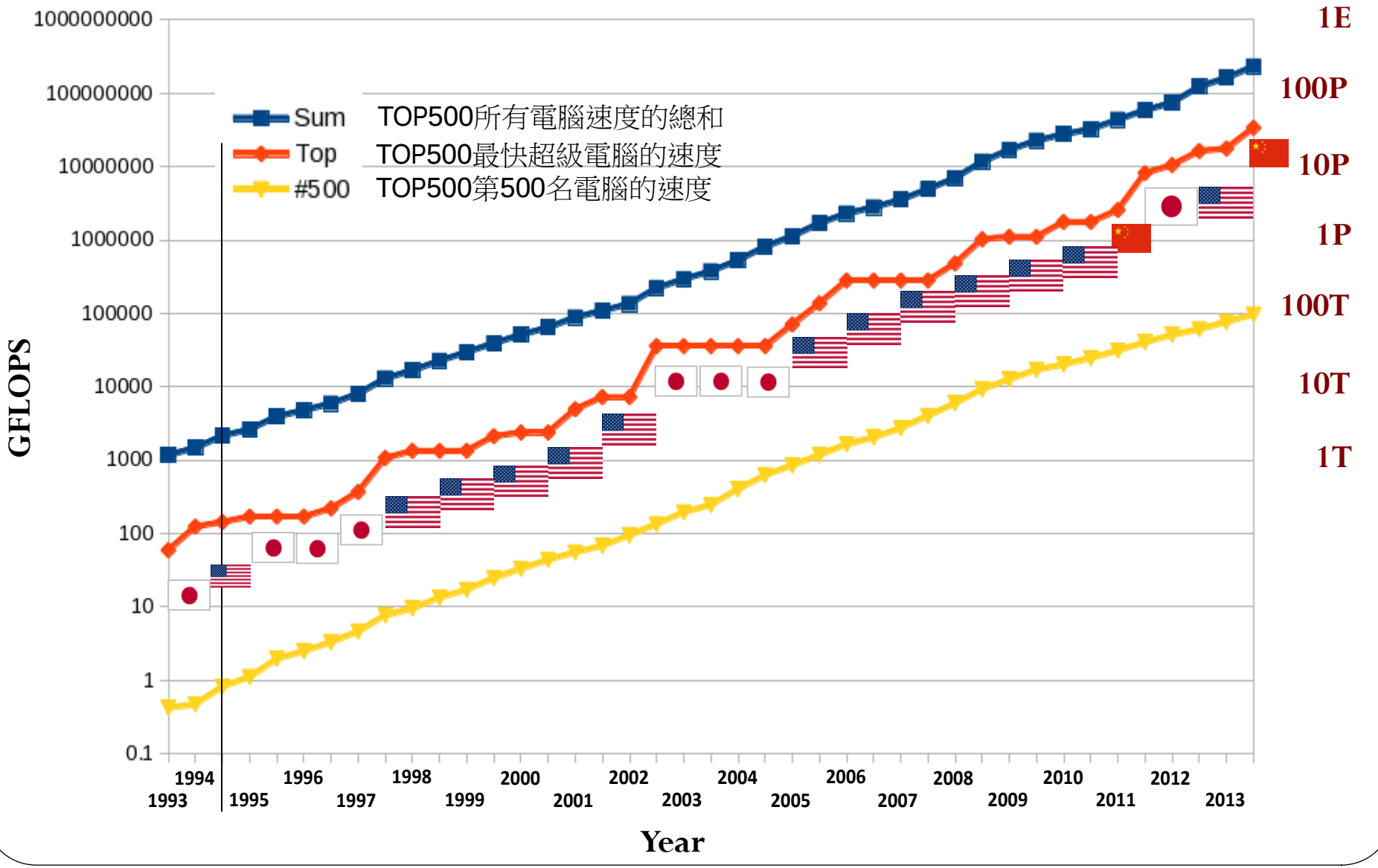
PC(2012)
100G FLOPS

x 100,000

K computer(2011)
10,000,000G FLOPS

Tremendous performance up

超級電腦歷年(1993年~2013年)速度統計

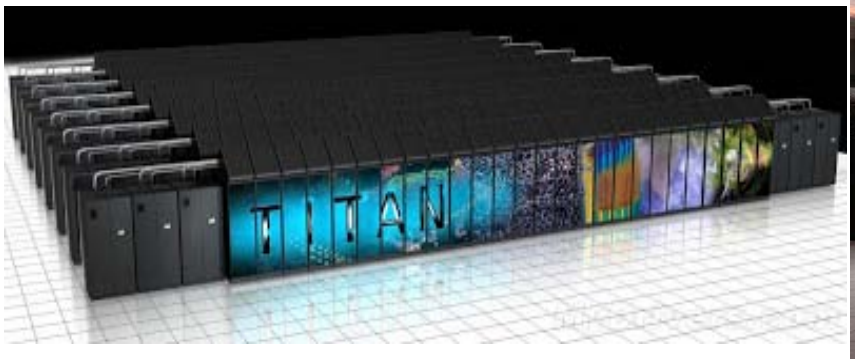


中國--天河二號 (Tianhe-2)

置放地點	中國大陸廣東省廣州市
架構	英特爾 Xeon E5-2692，英特爾 Xeon Phi 協處理器/運算加速卡
最大消耗功率	17.6MW （整機附帶散熱系統時爲 24MW）
佔地面積	720 平方米
內部存儲器	1,375TiB （1,000TiB 爲系統記憶體，375TiB 爲協處理器獨占）
外部存儲器	12.4PB
運算速率	54.9PFLOPS（理論峰值） 33.86PFLOPS（實際峰值）
造價	一億美元

美國--泰坦 (Titan)

置放地點	橡樹嶺國家實驗室
架構	18,688 顆 AMD Opteron 6274 16 核心中央處理器 18,688 顆 NVIDIA Tesla K20X 通用圖形處理器（運算加速卡） Cray Linux Environment
最大消耗功率	8.2 兆瓦
容積、佔地面積	404 平方米
內部存儲器	710TB（598TB 供中央處理器使用 、112TB 顯示記憶體）
外部存儲器	10PB，240GB/s 的輸出輸入頻寬
運算速率	17.59 petaFLOPS（LINPACK 基準效能測試） 27 petaFLOPS（理論峰值）
造價	9 千 7 百萬美元

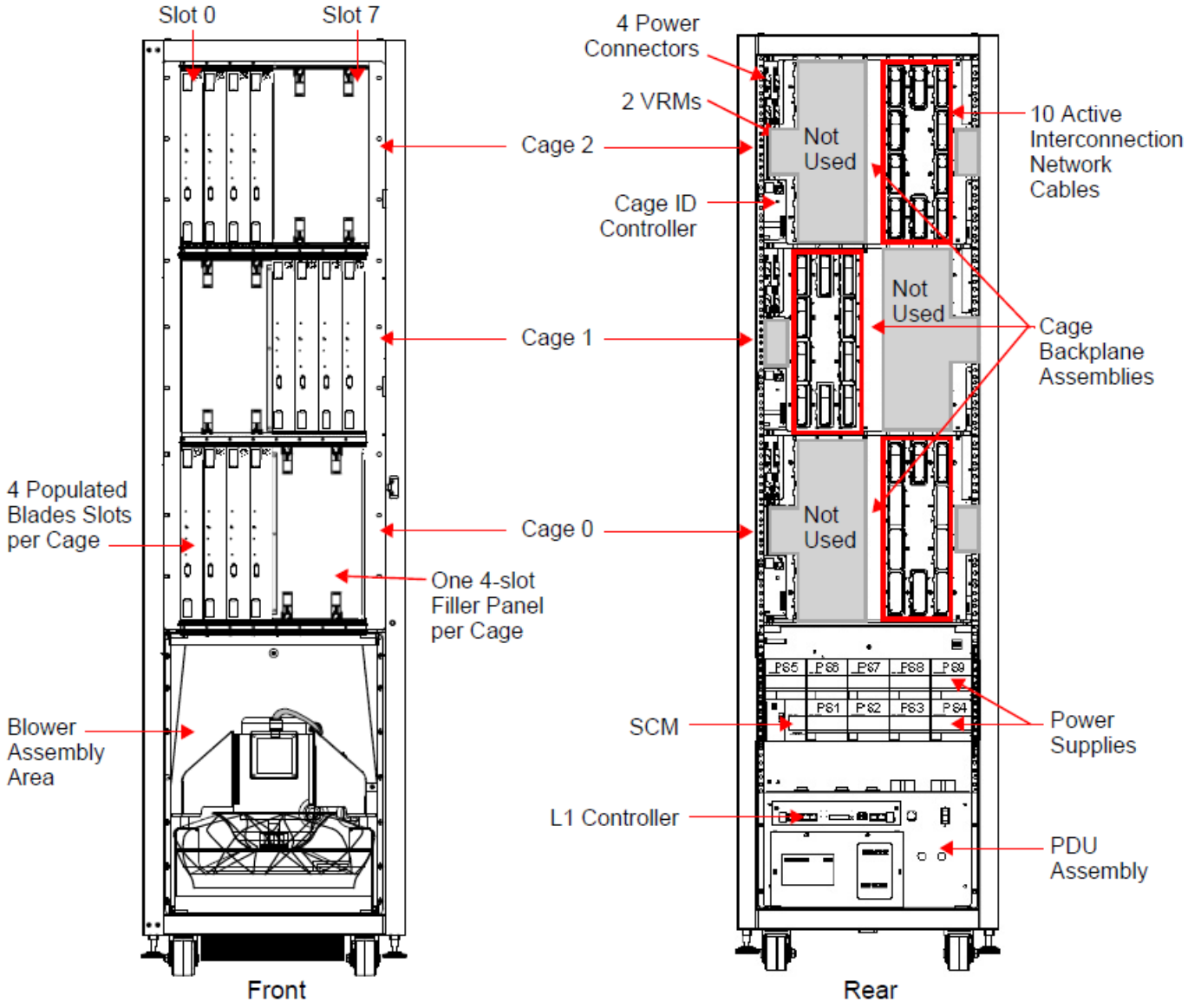


日本--京 (K Computer)

CRAY XE6m at Supercomputing Research Center

Attribute	Quantity
Peak Performance	4.45 TF
Server Rack count / Chassis count	1/2
Total Blade Count	8
Service Blades	2
1st type Compute Blades (Nodes)	2 (8 nodes)x AMD 2.4GHz IL-16 cores
1st type total Memory capacity	256GB (1GB/core)
2nd type Compute Blades (Nodes)	4 (16 nodes)x AMD 2.6GHz IS-6 cores
2nd type total Memory capacity	512GB (2.67/core)
Total Compute Core modules	448 Cores (256x cores in 2.4GHz + 192x cores in 2.6GHz)
Storage	Boot Raid with 20TB raw capacity
OS Software	Cray Linux Environment
PE Software	Cray Compiler & Scientific Libraries
Workload Manager	SLURM
Warranty	1 year HW/SW shared-support

Hardware Overview of CRAY XE6m

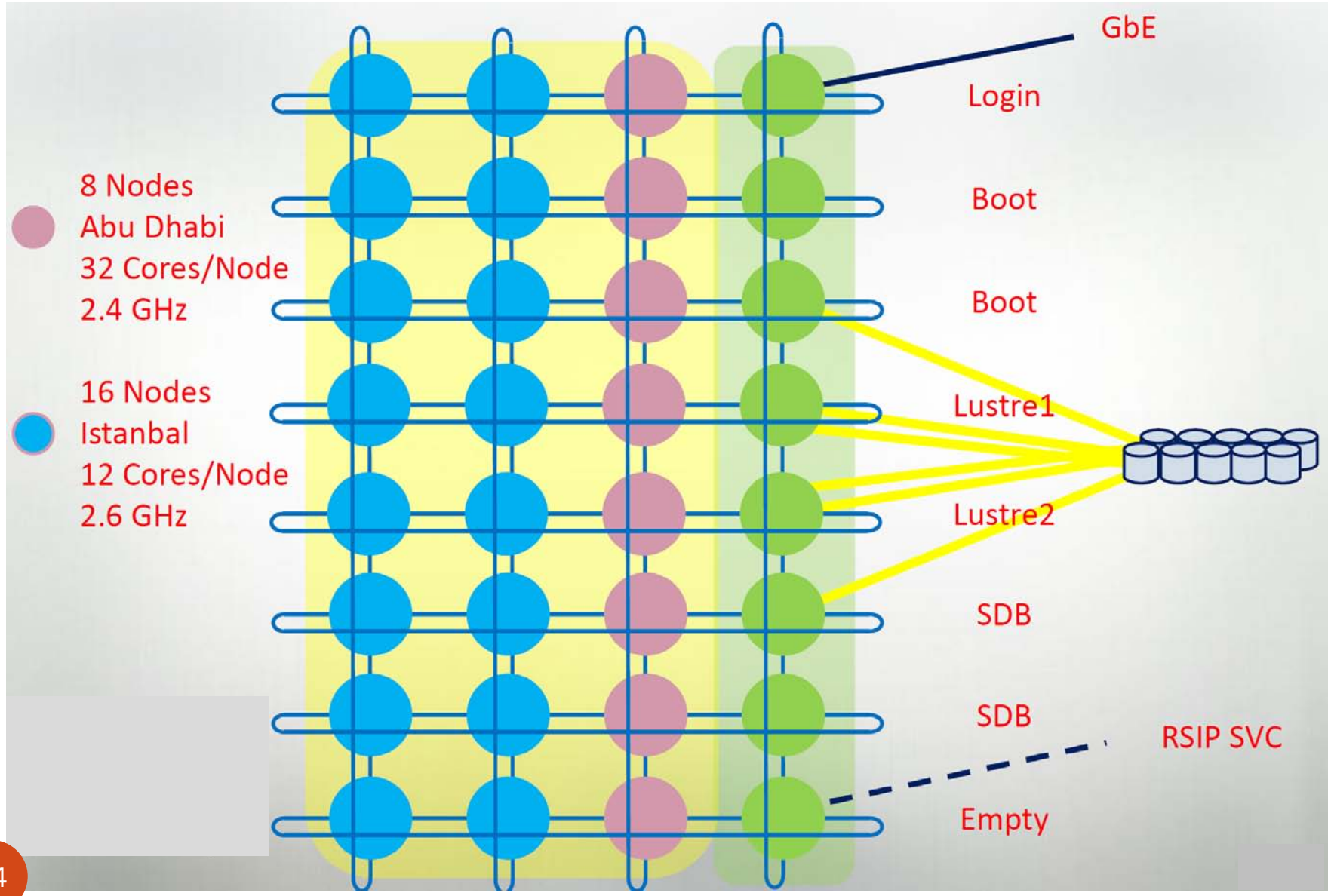


Specifications of CRAY XE6m

SYSTEM NAME	Cray-Blade I	Cray-Blade II
Performance	**2.45 Tflops**	**2.00 Tflops**
OPERATING SYSTEM	Cray Linux	Cray Linux
NUMBER OF NODES	8 nodes	16 nodes
MEMORY/NODE[GB]	64 GB	32 GB
TOTAL MEMORY[GB]	512 GB	512 GB
PROCESSOR CLOCK RATE	2.4 GHz	2.6 GHz
CORES PER NODE	16 cores	6 cores
TOTAL CPU CORES	128 cores	96 cores
NIC INTERFACE	2blade,YARC	4blade,YARC
STORAGE SYSTEM	Lustre/FC	Lustre/FC
STORAGE CAPACITY	20 TB / RAID5	20 TB / RAID5

½ Rack/Chassis for space required

Two Types of Processors at CRAY XE6m



Job Execution

- Resource scheduling : PBS Pro
- All jobs need to be summit by PBS Pro
- Current configuration:

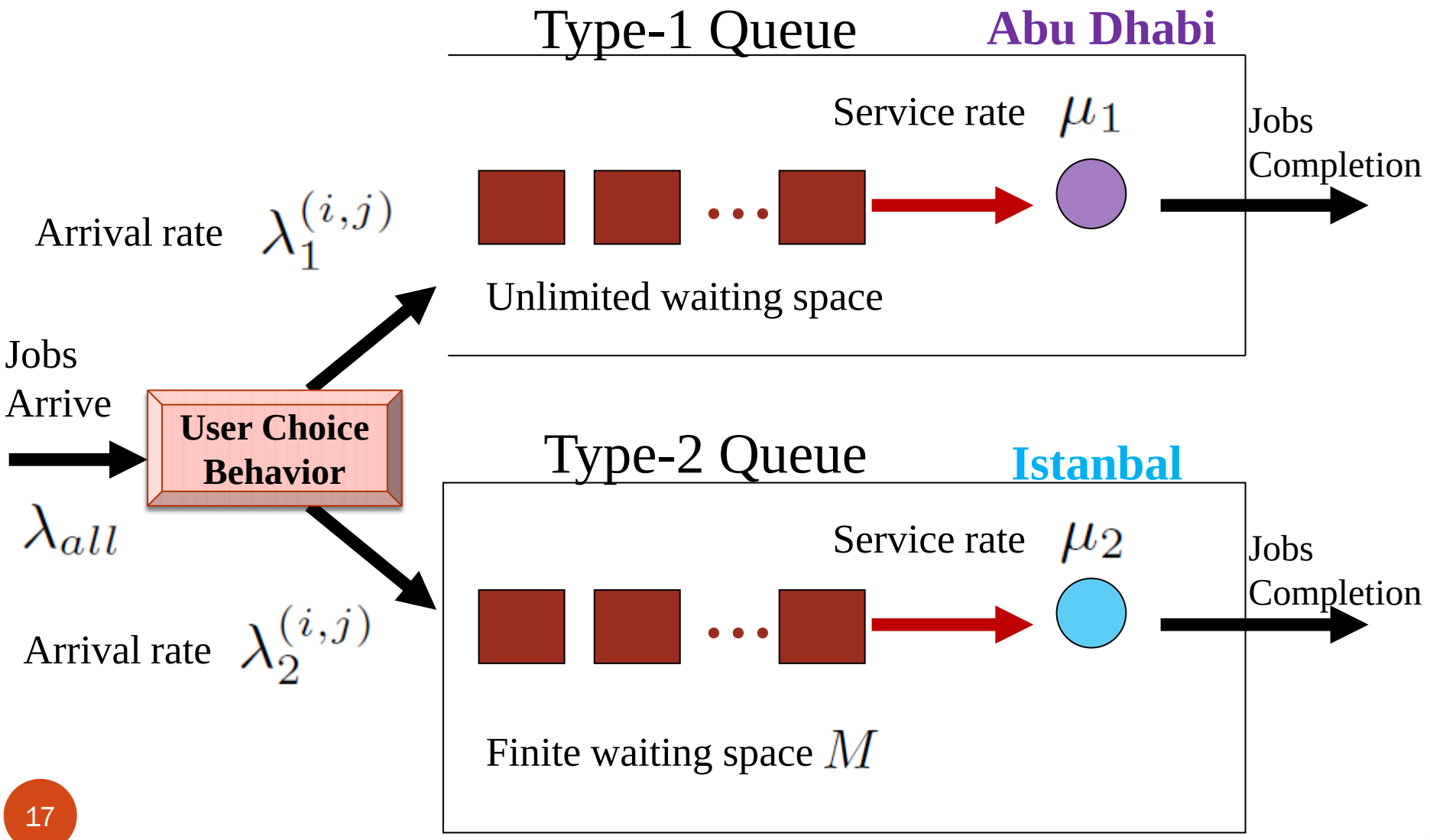
	CPU Spec	# of blade	# of node
Type-1	Abu Dhabi 16-core 2.4GHz	2	8
Type-2	Istanbul 6-core 2.6GHz	4	16

- How to summit job:
  - module load pbs
  - vi *script.sh* (in the location of executable file)
  - chmod u+x *script.sh*
  - qsub *script.sh*
  - apstat or qstat (check if the job summit successfully)
  - xtnodestat (see what node using)

Outline

- Introduction to Supercomputer CRAY XE6m at NCKU
- A Queueing Model for Two Heterogeneous Processors with a Finite Buffer
- A Computing Approach—K-matrix Based Algorithm
- Numerical Comparisons
- Conclusions and Future Works

A Two-type Service Queueing Model



Stability Condition

Proposition 1. *The service system reaches the steady state if*

$$\mu_1 > \frac{\left(1-\frac{\lambda_{all}}{\mu_2}\right)\left(\frac{\lambda_{all}}{\mu_2}\right)^M \lambda_{all}}{1-\left(\frac{\lambda_{all}}{\mu_2}\right)^{M+1}}. \qquad (1)$$

Note that, when $M \longrightarrow \infty$, the condition (1) is reduced to more intuitive stability conditions.

Proposition 2. *As* $M \longrightarrow \infty$, *we have*

(i.) the stability condition $\mu_1 > 0$ *if* $\lambda_{all}/\mu_2 \leq 1$;

(ii.) the stability condition $\mu_1 + \mu_2 > \lambda_{all}$ *if* $\lambda_{all}/\mu_2 > 1$.

State Dependent Quasi-Birth-and-Death Process

Under the stability condition, the stationary probability vector is defined as

$$\boldsymbol{\pi}_n = [\pi_{n0}, \pi_{n1}, \ldots, \pi_{nM}], \quad \text{for } n=0,1,2,\ldots$$

$$\mathbf{Q} = \begin{bmatrix}
\mathbf{C}_{0,0} & \mathbf{A}_{0,1} & & & & & & \\
\mathbf{D} & \mathbf{C}_{1,1} & \mathbf{A}_{1,2} & & & & & \\
 & \mathbf{D} & \mathbf{C}_{2,2} & \mathbf{A}_{2,3} & & & & \\
 & & \ddots & \ddots & \ddots & & & \\
 & & & \mathbf{D} & \mathbf{C}_{M,M} & \mathbf{A}_{M,M+1} & & \\
 & & & & \mathbf{D} & \mathbf{C} & \mathbf{A} & \\
 & & & & & \ddots & \ddots & \ddots
\end{bmatrix}$$

Special Structure of Submatrices in Matrix **Q**

$$\mathbf{C}_{0,0} = \begin{bmatrix} -\lambda_{all} & & & \\ \mu_2 & -(\lambda_{all}+\mu_2) & & \\ & \ddots & \ddots & \\ & & \mu_2 & -(\lambda_{all}+\mu_2) \end{bmatrix} \quad \mathbf{A}_{0,1} = \begin{bmatrix} \lambda_{all} & & \\ & \ddots & \\ & & \lambda_{all} \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_1 \end{bmatrix}$$

$$\Theta = \lambda_{all} + \mu_2 + \mu_1$$

$$\mathbf{C}_{i,i} = \begin{bmatrix} -(\mu_1+\lambda_{all}) & \lambda_2^{(i,0)} & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2 & -\Theta & \lambda_2^{(i,1)} & 0 & \cdots & \cdots & \vdots \\ 0 & \mu_2 & -\Theta & \lambda_2^{(i,2)} & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_2^{(i,i-2)} & 0 \\ \vdots & \cdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_2^{(i,i-1)} \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_2 & -\Theta \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \lambda_{all} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} -(\mu_1+\lambda_{all}) & \lambda_{all} & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2 & -\Theta & \lambda_{all} & 0 & \cdots & \cdots & \vdots \\ 0 & \mu_2 & -\Theta & \lambda_{all} & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_{all} & 0 \\ \vdots & \cdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_{all} \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_2 & -\Theta \end{bmatrix} \quad \mathbf{A}_{i,i+1} = \begin{bmatrix} \lambda_1^{(i,0)} & & & & & \\ & \ddots & & & & \\ & & \lambda_1^{(i,i-1)} & & & \\ & & & \lambda_{all} & & \\ & & & & \ddots & \\ & & & & & \lambda_{all} \end{bmatrix}$$

Matrix Geometric Solution

From $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ and $\boldsymbol{\pi}_{n+1} = \boldsymbol{\pi}_n\mathbf{R}$, $n \geq M$,

where $\mathbf{R}$ is the rate matrix, the probability vector $\boldsymbol{\pi}_n$ can be obtained by solving

$$\boldsymbol{\pi}_0\mathbf{C}_{0,0} + \boldsymbol{\pi}_1\mathbf{D} = \mathbf{0},$$

$$\boldsymbol{\pi}_0\mathbf{A}_{0,1} + \boldsymbol{\pi}_1\mathbf{C}_{1,1} + \boldsymbol{\pi}_2\mathbf{D} = \mathbf{0},$$

$$\boldsymbol{\pi}_1\mathbf{A}_{1,2} + \boldsymbol{\pi}_2\mathbf{C}_{2,2} + \boldsymbol{\pi}_3\mathbf{D} = \mathbf{0},$$

$$\vdots$$

$$\boldsymbol{\pi}_{M-2}\mathbf{A}_{M-2,M-1} + \boldsymbol{\pi}_{M-1}\mathbf{C}_{M-1,M-1} + \boldsymbol{\pi}_M\mathbf{D} = \mathbf{0},$$

$$\boldsymbol{\pi}_{M-1}\mathbf{A}_{M-1,M} + \boldsymbol{\pi}_M(\mathbf{C}_{M,M} + \mathbf{RD}) = \mathbf{0},$$

$$\boldsymbol{\pi}_0\mathbf{1} + \boldsymbol{\pi}_1\mathbf{1} + \cdots + \boldsymbol{\pi}_M(\mathbf{I} - \mathbf{R})^{-1}\mathbf{1} = 1.$$

Outline

- Introduction to Supercomputer CRAY XE6m at NCKU
- A Queueing Model for Two Heterogeneous Processors with a Finite Buffer
- A Computing Approach—K-matrix Based Algorithm
- Numerical Comparisons
- Conclusions and Future Works

A Computing Approach

We consider an $(M+1)\times(M+1)$ matrix

$$\mathbf{K} = -[\mathbf{C}+\mathbf{\Phi}]\mathbf{D}^{-1},$$

$$\mathbf{C} = \begin{bmatrix} -(\mu_1+\lambda_{all}) & \lambda_{all} & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2 & -\Theta & \lambda_{all} & 0 & \cdots & \cdots & \vdots \\ 0 & \mu_2 & -\Theta & \lambda_{all} & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_{all} & 0 \\ \vdots & \cdots & \cdots & 0 & \mu_2 & -\Theta & \lambda_{all} \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_2 & -\Theta \end{bmatrix}$$

$$\mathbf{\Phi} \triangleq \begin{bmatrix} 0 & \cdots & 0 & \mu_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \mu_1 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_1 \end{bmatrix}$$

$$\Theta = \lambda_{all} + \mu_2 + \mu_1$$

Properties of **K** Matrix

Proposition 3. *The eigenvalues of matrix* **K** *are positive.*

Proposition 4. *There exists a unique eigenvalue* σ *of* **K** *between 0 and 1.*

Proposition 5. *Under the stability condition, we have*

$$\sigma(M) = \frac{\lambda_{all}}{\mu_1 + \mu_2} + o(M),$$

which implies

$$\lim_{M \to \infty} \sigma(M) = \frac{\lambda_{all}}{\mu_1 + \mu_2}.$$

K-matrix Based Algorithm

We suppose that $0 < \sigma < 1$ is an eigenvalue of $\mathbf{K}$, and denote $\mathbf{v}^T = (v(1), v(2), \cdots, v(M+1))$ as the corresponding eigenvector. Set

$$\mathbf{R}' = h \begin{bmatrix} \mathbf{0} \\ \mathbf{v}^T \end{bmatrix},$$

where $h = \sigma / v(M+1)$. Because of $\mathbf{R}'^2 = \sigma \mathbf{R}' = \mathbf{R}'\mathbf{K}$, it can be derived that $\mathbf{R}'$ is a solution of $\mathbf{A} + X\mathbf{C} + X^2\mathbf{D} = \mathbf{0}$.

Proposition 6. The matrix $\mathbf{R}'$ is a solution of $\mathbf{A} + X\mathbf{C} + X^2\mathbf{D} = \mathbf{0}$.

K-matrix Based Algorithm

Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$, where

$$\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_0(0), \boldsymbol{\pi}_0(1), \ldots, \boldsymbol{\pi}_0(M)).$$

We define the matrix $\mathbf{T}_i$ as follows

$$\begin{aligned} \mathbf{T}_0 &= \mathbf{I}, \\ \mathbf{T}_1 &= -\mathbf{C}_{0,0}\mathbf{D}^{-1}, \\ \mathbf{T}_i &= -(\mathbf{T}_{i-2}\mathbf{A}_{i-2,i-1} + \mathbf{T}_{i-1}\mathbf{C}_{i-1,i-1})\mathbf{D}^{-1}, \end{aligned}$$

for $2 \leq i \leq M$.

K-matrix Based Algorithm

Proposition 7. *The stationary probability* $\boldsymbol{\pi}_0(0)$ *satisfies the following two equations:*

(i.) $\boldsymbol{\pi}_0(\mathbf{T}_{M-1}\mathbf{A}_{M-1,M} + \mathbf{T}_M(\mathbf{C} + \mathbf{R}'\mathbf{D})) = \mathbf{0}$, *where*

$$\mathbf{R}' = h \begin{bmatrix} \mathbf{0} \\ \mathbf{v}^T \end{bmatrix}, \quad \mathbf{v}^T\mathbf{1} = 1;$$

(ii.)

$$\boldsymbol{\pi}_0 \left[\sum_{i=0}^{M-1} \mathbf{T}_i\mathbf{1} + \mathbf{T}_M \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \bar{h} \end{bmatrix} \right] = 1,$$

where

$$\bar{h} = 1 + \frac{\lambda_{all}}{\mu_1(1-\sigma)}$$

and $\boldsymbol{\pi}_i = \boldsymbol{\pi}_0\mathbf{T}_i$, *for* $0 \le i \le M$.

K-matrix Based Algorithm

Proposition 8. *The solution vector* $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots)$ *of* $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ *can be obtained from*

$$\begin{aligned} \boldsymbol{\pi}_0(i) &= \boldsymbol{\pi}_0(0)\mathbf{T}_i, & \forall 0 \leq i \leq M, \\ \boldsymbol{\pi}_{M+k} &= \boldsymbol{\pi}_M \sigma^{k-1}\mathbf{R}', & \forall 1 \leq k. \end{aligned}$$

K-matrix Based Algorithm

Step 1. Set $\mathbf{K} = -[\mathbf{C} + \mathbf{\Phi}]\mu_1^{-1}$.

Step 2. Find an eigenvalue σ of $\mathbf{K}$ which is less than one, and has a corresponding right eigenvector $\mathbf{v}$, where $\mathbf{v}^T\mathbf{1} = 1$.

Step 3. Define $\mathbf{R}' = h \begin{bmatrix} \mathbf{0} \\ \mathbf{v}^T \end{bmatrix}$, where $h = \sigma/v(M+1)$.

Step 4. Construct matrices $\mathbf{T}_0 = \mathbf{I}$, $\mathbf{T}_1 = -\mathbf{C}_{0,0}\mathbf{D}^{-1}$ and $\mathbf{T}_i = -(\mathbf{T}_{i-2}\mathbf{A}_{i-2,i-1} + \mathbf{T}_{i-1}\mathbf{C}_{i-1,i-1})\mathbf{D}^{-1}$ for $2 \le i \le M$.

Step 5. Determine $\boldsymbol{\pi}_0$ by solving

$$\boldsymbol{\pi}_0(\mathbf{T}_{M-1}\mathbf{A}_{M-1,M} + \mathbf{T}_M(\mathbf{C} + \mathbf{R}'\mathbf{D})) = \mathbf{0},$$

$$\boldsymbol{\pi}_0 \left[\sum_{i=0}^{M-1} \mathbf{T}_i\mathbf{1} + \mathbf{T}_M \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \bar{h} \end{bmatrix} \right] = 1,$$

and $\boldsymbol{\pi}_0(i) = \boldsymbol{\pi}_0(0)\mathbf{T}_i$, for $0 \le i \le M$.

Step 6. From $\boldsymbol{\pi}_{M+k} = \boldsymbol{\pi}_M\sigma^{k-1}\mathbf{R}'$, for $k \ge 1$, we obtain $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$.

Outline

- Introduction to Supercomputer CRAY XE6m at NCKU
- A Queueing Model for Two Heterogeneous Processors with a Finite Buffer
- A Computing Approach—K-matrix Based Algorithm
- Numerical Comparisons
- Conclusions and Future Works

Numerical Comparisons

➢ We compare the numerical performance under two computing approaches, i.e., Matrix-Geometric method versus **K**-matrix based algorithm.

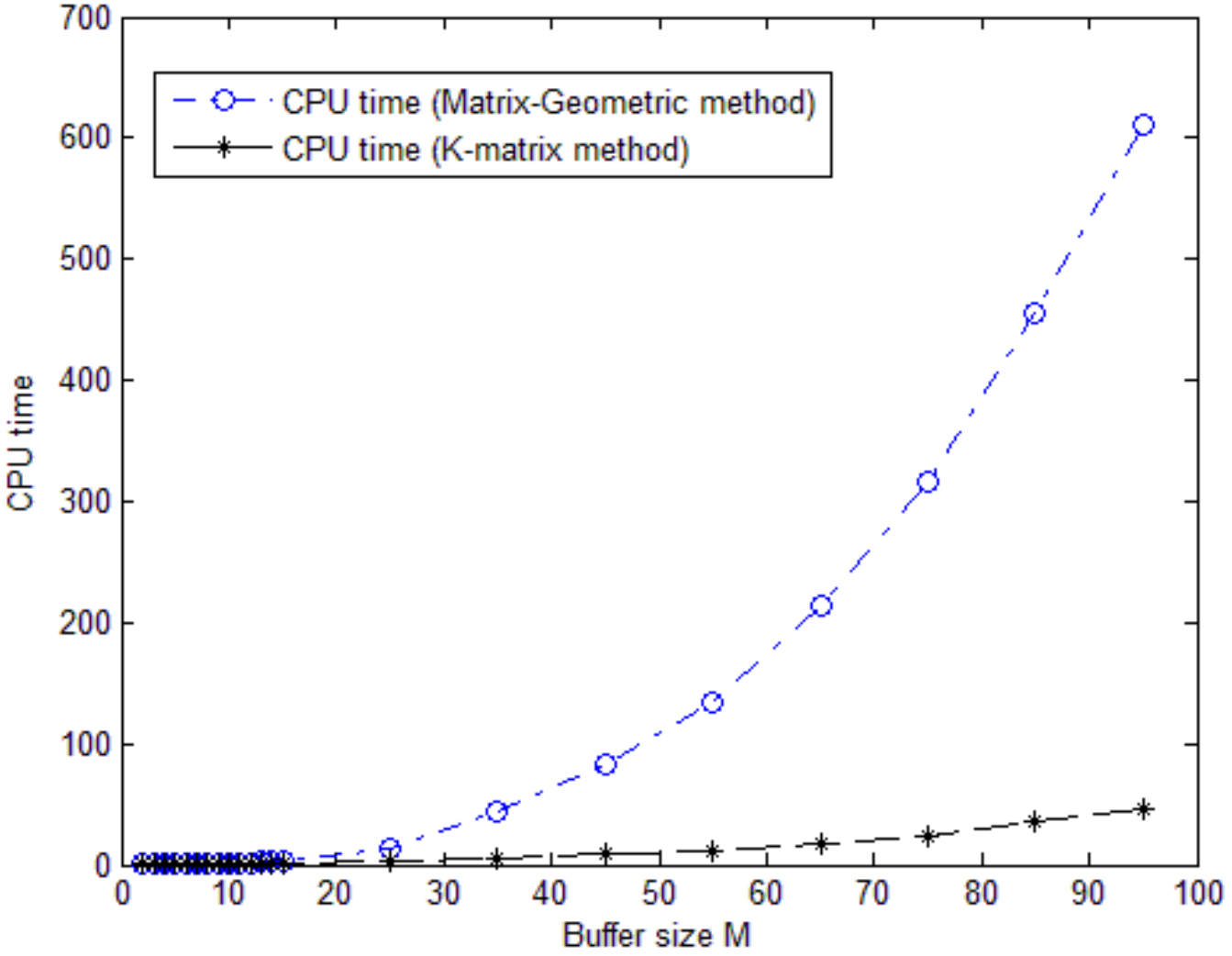
➢ We compile MATLAB program on the PC platform with Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz and 32 GB RAM.

➢ Given parameters $\lambda_{all} = 1$ **and** $\mu_1 = \mu_2 = 0.6$.

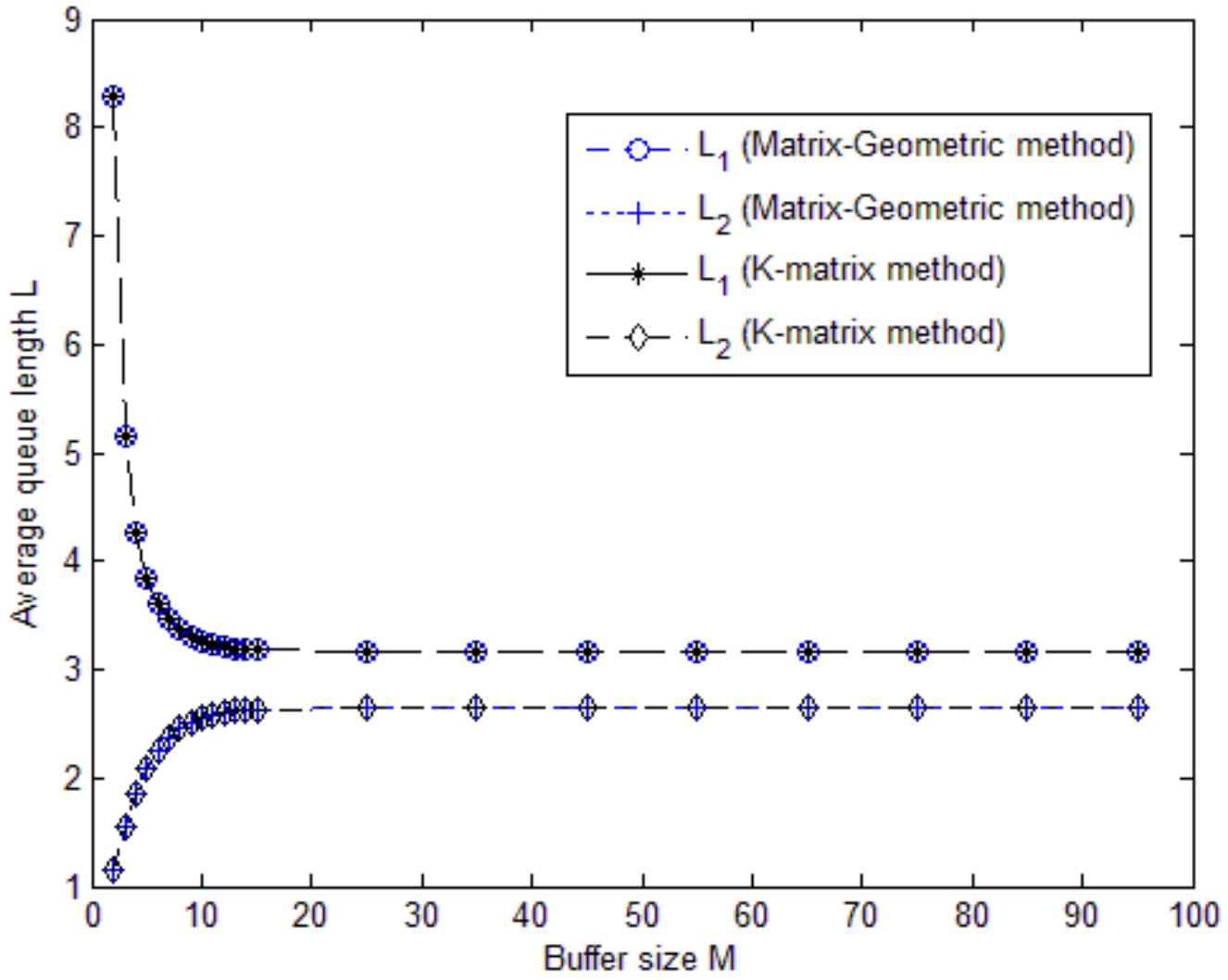
Numerical Comparisons between Matrix-Geometric Method and **K**-matrix Method

	Matrix-Geometric Method			**K**-matrix Method		
M	L_1	L_2	CPU Time	L_1	L_2	CPU Time
5	3.6126	2.0068	0.3588	3.9458	2.1157	0.1560
15	3.1416	2.6050	0.7956	3.1848	2.6386	0.7020
25	3.1606	2.6475	2.0904	3.1634	2.6500	0.4212
35	3.1626	2.6501	12.6205	3.1627	2.6502	0.7644
45	3.1627	2.6502	24.8198	3.1627	2.6502	2.0436
55	3.1627	2.6502	42.8535	3.1627	2.6502	3.5256
65	3.1627	2.6502	86.7054	3.1627	2.6502	6.1152
75	3.1627	2.6502	159.7138	3.1627	2.6502	12.4333
85	3.1627	2.6502	305.6372	3.1627	2.6502	23.9306
95	3.1627	2.6502	542.7587	3.1627	2.6502	43.1499

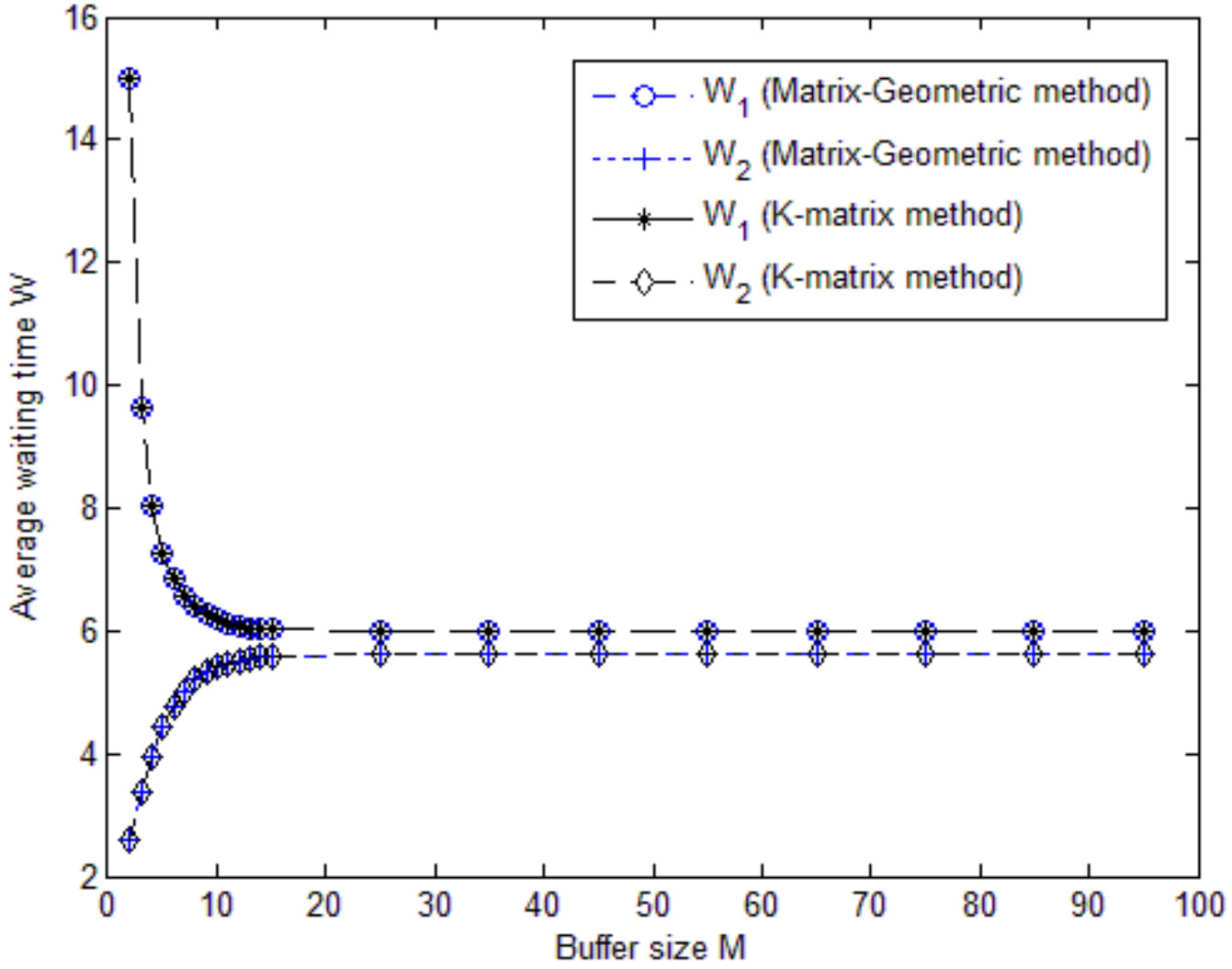
CPU Time versus Buffer Size



Average System Size versus Buffer Size



Average Waiting Time versus Buffer Size



Managerial Insights

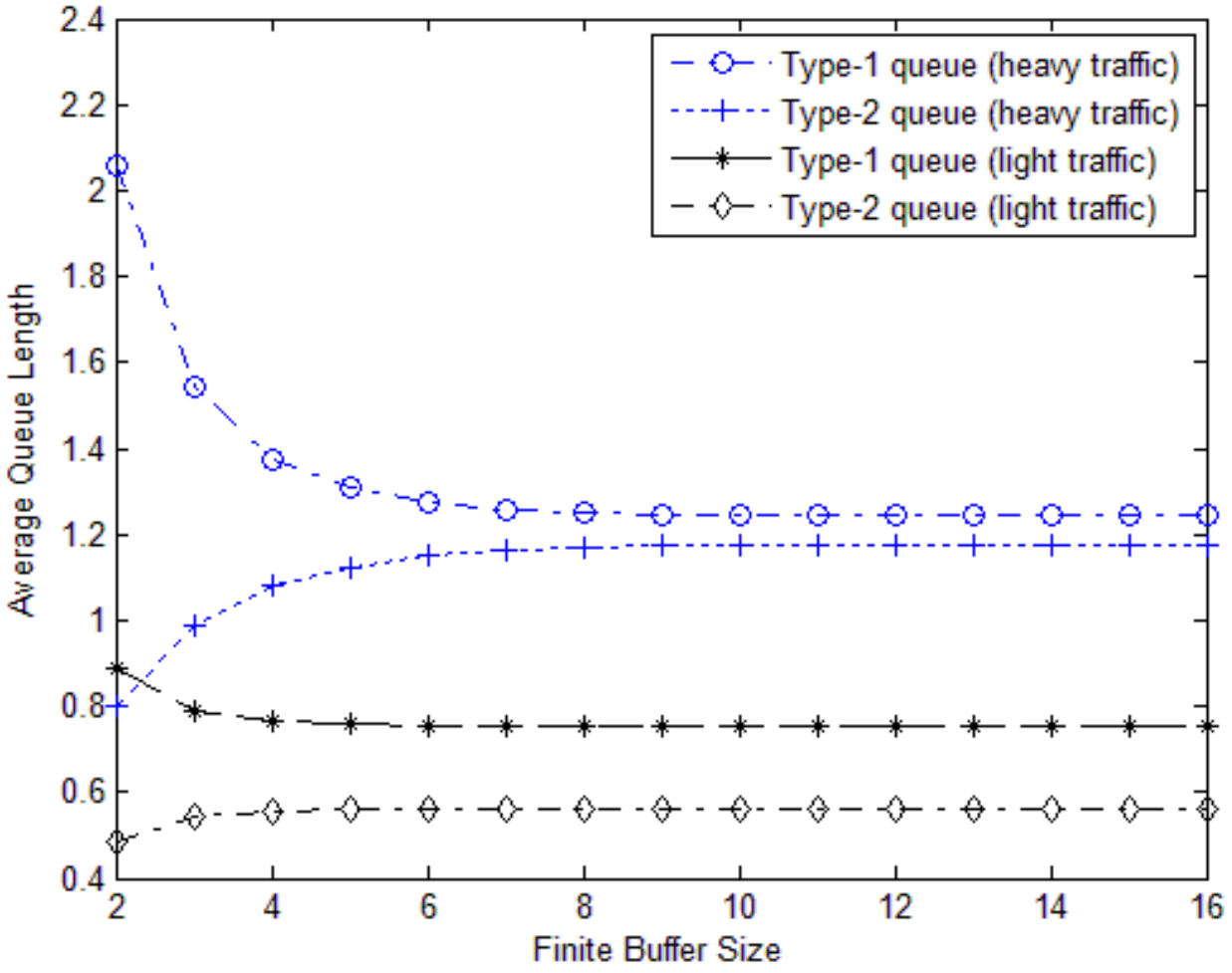
- In this numerical experiments, we consider two scenarios: heavy traffic load and light traffic load.

- For the heavy traffic, set $\lambda = 0.9,\ \mu_1 = 0.6$ and $\mu_2 = 0.8$.

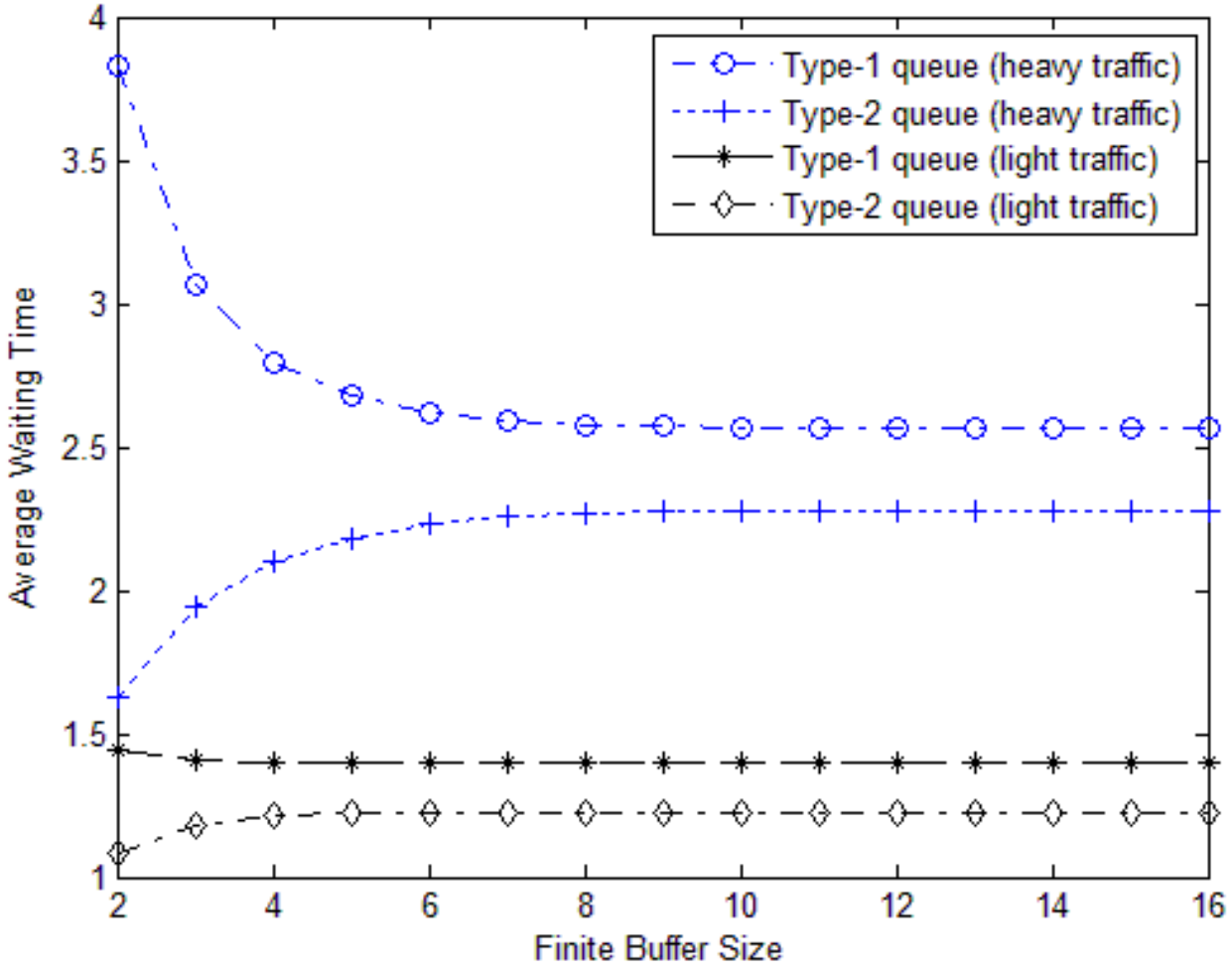
- For the light traffic, set $\lambda = 0.65,\ \mu_1 = 0.6$ and $\mu_2 = 0.8$.

- As the finite buffer size M varies from 2 to 16, we demonstrate the effect of finite buffer control on the system performances, i.e., average queue length and average waiting time.

Average Queue Length versus Buffer Size



Average Waiting Time versus Buffer Size



Outline

- Introduction to Supercomputer CRAY XE6m at NCKU
- A Queueing Model for Two Heterogeneous Processors with a Finite Buffer
- A Computing Approach—K-matrix Based Algorithm
- Numerical Comparisons
- Conclusions and Future Works

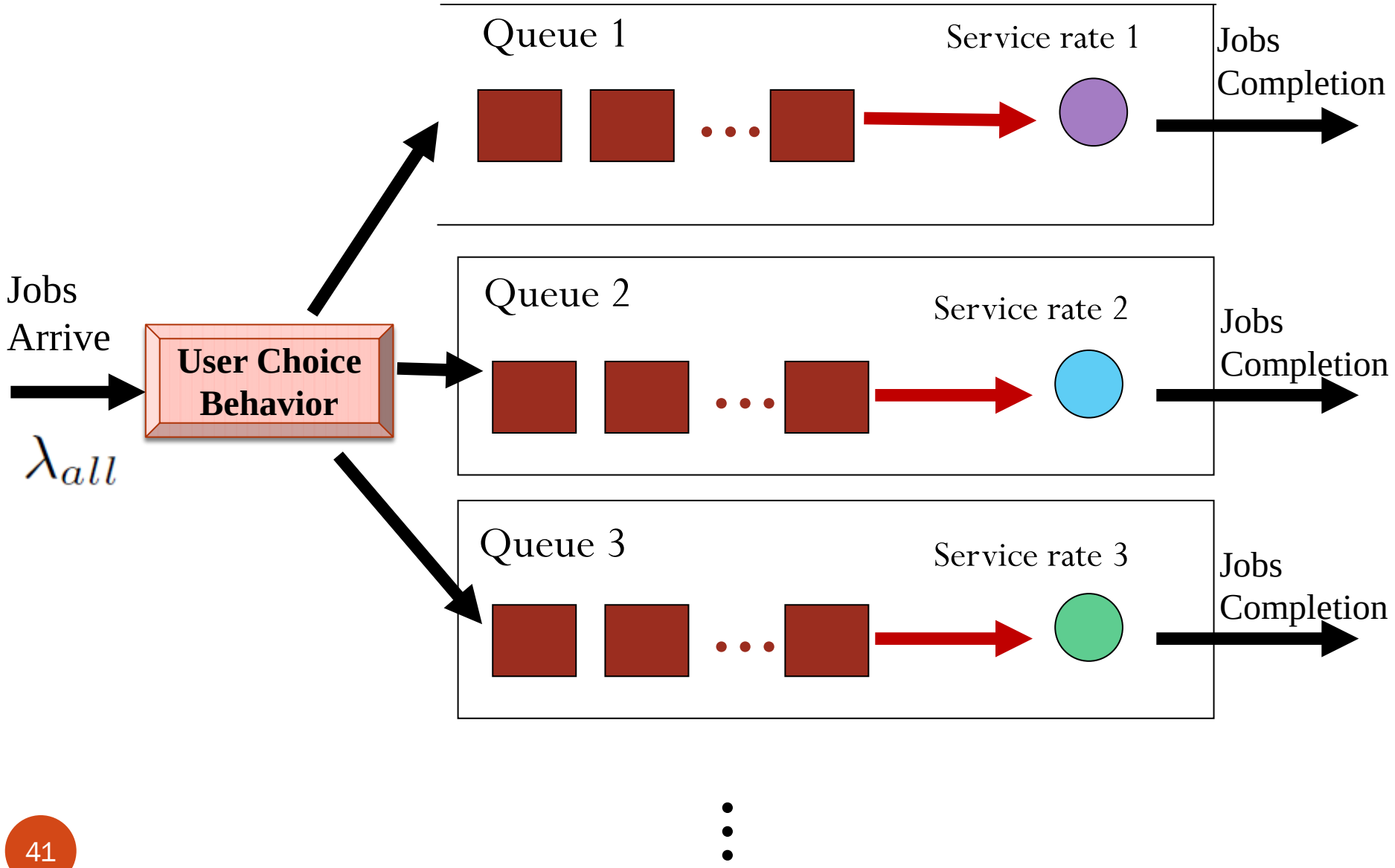
Conclusions

- For solving this two-type heterogeneous system, we find that the proposed **K**-matrix method is a more efficient algorithm than Geometric-Matrix method, especially when the buffer size is large.

- The proposed approach depends only on an eigenvalue of matrix **K** and its corresponding eigenvector.

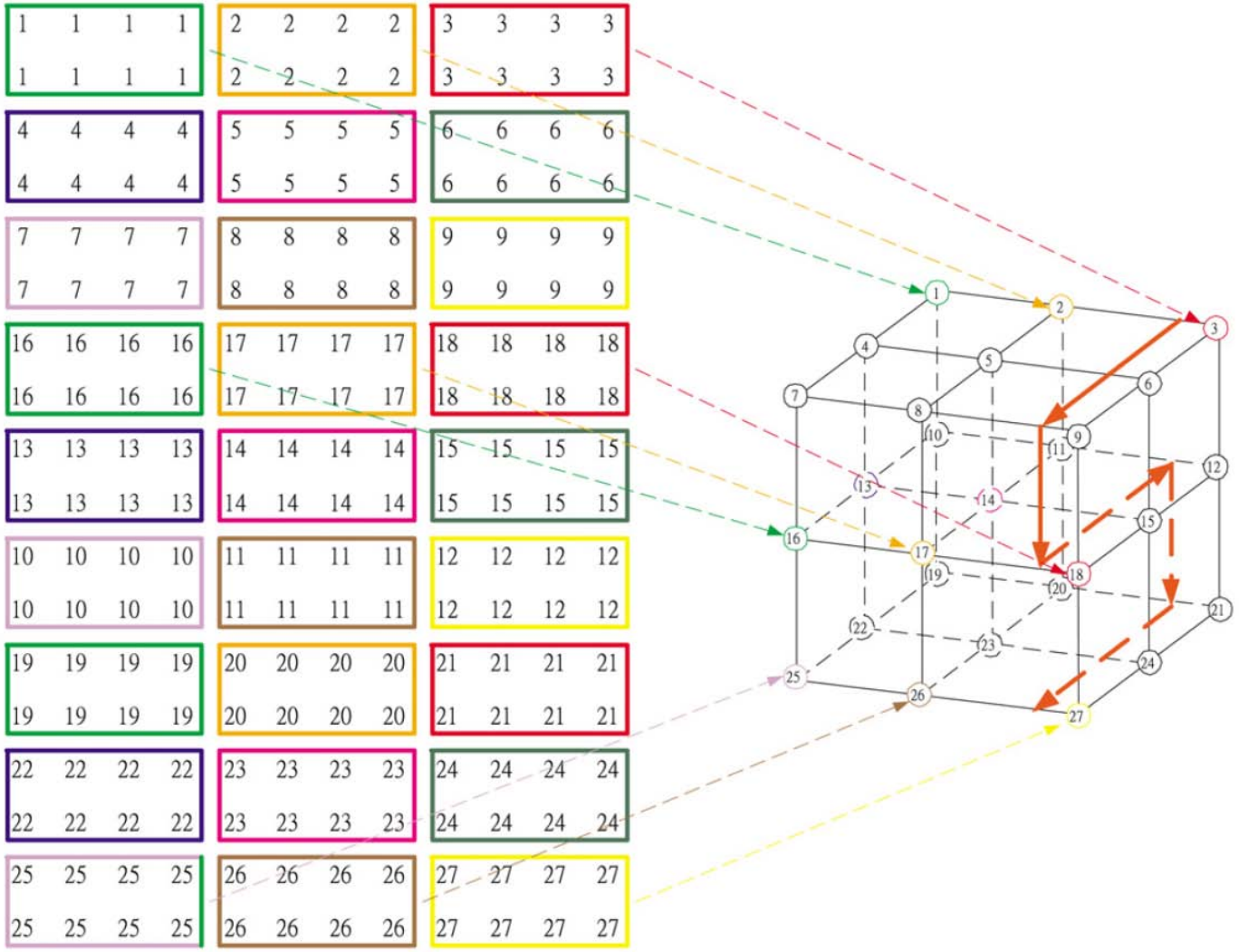
- The computational complexity of **K**-matrix method is comparatively simple and low because it only solve the vector π_0 and the remaining probabilities are attained by substitution.

Future Works



Future Works

Task Mapping Problem on Supercomputers



Reference

- Bucur, A.I. and Epema, D.H., The influence of the structure and sizes of jobs on the performance of co-allocation, *Lecture Notes in Computer Science*, vol. 1911, pp. 154-173, 2000.
- Chlamtac, I. Kienzle, M.G., and Szabo, C., Characterizing the behaviour of high-speed interconnection systems with distributed control, *Telecommunication Systems*, vol. 6, pp. 91-115, 1996.
- Downey, A.B., Using queue time predictions for processor allocation, *Proceedings of the 3rd Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 35-57, 1997.
- Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., and Wong, P., Theory and practice in parallel job scheduling, *Proceedings of the 3rd Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 1-34, 1997.

Reference

- Kerbyson, D.J., Barker, K.J., Vishnu, A., and Hoisie, A., A performance comparison of current HPC systems: Blue Gene/Q, Cray XE6 and InfiniBand systems, *Future Generation Computer Systems*, vol. 30, pp. 291-304, 2014.
- von Laszewski, G., A loosely coupled metacomputer: co-operating job submissions across multiple supercomputing sites, *Concurrency Practice and Experience*, vol. 11, pp. 933-948, 1999.
- Luh, H., Zhang, Z., and Wang, C.H., A computing approach to two competing services with a finite buffer effect, *Proceedings of the 8th International Conference on Queueing Theory and Network Applications (QTNA2013)*, pp. 15-21, 2013.
- Neuts, M.F., *Matrix-Geometric Solutions in Stochastic Models*. The John Hopkins University Press, 1981.

Reference

- Piro, R.M., Guarise, A., Patania, G., and Werbrouck, A., Using historical accounting information to predict the resource usage of grid jobs, *Future Generation Computer Systems*, vol. 25, pp. 499-510, 2009.
- Schroeder, B. and Harchol-Balter, M., Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness, *Cluster Computing*, vol. 7, pp. 151-161, 2004.
- Chia-Hung Wang, Yu-Tin Chen, and Chi-Chuan Hwang, A Queueing Analysis for Job Assignment on Two-Type Heterogeneous Supercomputer System, *Proceedings of 2015 International Conference on Computer Information Systems and Industrial Applications (CISIA2015)*, pp. 375--378, Bangkok, Thailand, June 28-29, 2015.
- Tang, W., Ren, D., Lan, Z., and Desai, N., Toward balanced and sustainable job scheduling for production supercomputers, *Parallel Computing*, vol. 39, pp. 753-768, 2013.

Thank You for Your Attention!

Should there is any questions/comments, please feel free to contact me.

王 嘉 宏 博士

Dr. Chia-Hung Wang

College of Information Science and Engineering,
Fujian University of Technology,
Fuzhou City, Fujian Province 350118, P.R. China

E-mail: jhwang728@hotmail.com