งานที่มอบหมาย: ทดสอบประสิทธิภาพเครื่องและเขียนรายงานการทดสอบ

กลุ่มละ 3 คน และ ส่งวันที่ 22 ก.ค. 2555

รายละเอียด:

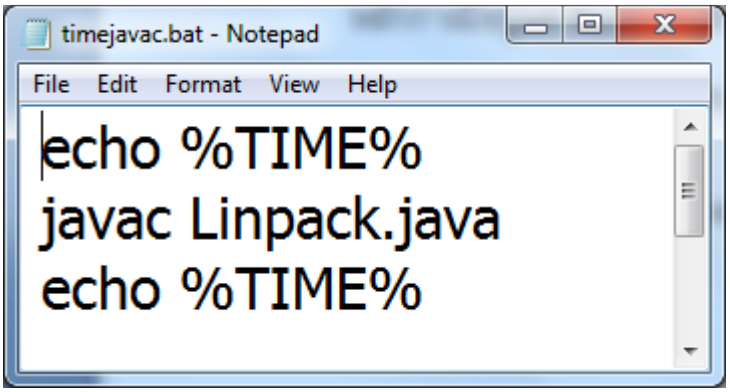
ทดสอบประสิทธิภาพของคอมพิวเตอร์ 3 เครื่อง (ถ้าเป็นไปได้ ให้เลือกเครื่องคนละแบบกัน) โดยใช้ ซอฟต์แวร์อย่างน้อย 4 อย่าง: (1) Java Compiler, (2) Linpack, และอีก 2 ซอฟต์แวร์ที่คุณเลือก

รายงานระบุรายละเอียดของคอมพิวเตอร์ที่ทดสอบทั้งฮาร์ดแวร์ ระบบปฏิบัติการ และซอฟต์แวร์ที่ใช้ โดยคำนึงถึง reproducibility. การทดสอบแต่ละรายการใช้ทำซ้ำไม่น้อยกว่า 3 ครั้งเพื่อลดผลของความแปรผัน โดยใช้ค่าเฉลี่ย และสรุปผลการทดสอบเป็นค่าเฉลี่ยตามน้ำหนักโดยให้น้ำหนักของผลจาก Java Compiler และ Linpack แต่ละอันเป็น 0.1 และคุณเลือกน้ำหนักที่จะให้กับ 2 ซอฟต์แวร์ที่คุณเลือกเอง (แต่ผลรวมของน้ำหนักทั้งหมด เป็น 1 เช่น ตามตัวอย่างข้างล่าง น้ำหนักของ Java Compiler, Linpack, Neural Network Classifier, และ Latex เป็น 0.1, 0.1, 0.3, และ 0.5 ตามลำดับ ซึ่งผมเลือกให้น้ำหนัก Latex มากกว่า Neural Network Classifier เนื่องจากผมใช้ Latex บ่อยกว่า Neural Network Classifier และ ผลรวมของน้ำหนักทั้งหมด 0.1 + 0.1 + 0.3 + 0.5 = 1)

ตัวอย่างการวัดประสิทธิภาพของคอมพิวเตอร์

หลักการมีอยู่แค่ จับเวลาที่ใช้รันโปรแกรม ตัวอย่างนี้เราจะวัดประสิทธิภาพของคอมพิวเตอร์จาก 4 ซอฟต์แวร์: Java Compiler, Linpack, Neural Network Image Classifier, และ Latex.

เตรียมการทดสอบ: เพื่อทดสอบเวลาที่ใช้รัน Java Compiler, ผมเขียน batch file ขึ้นมาด้วยโค้ดดังรูปที่ 1



รูป 1 **Batch file: timejavac.bat ที่เขียนบน notepad**

บรรทัดแรก และ สุดท้าย คำสั่ง echo %TIME% เป็น DOS COMMAND ให้แสดงเวลา

คำสั่งที่อยู่ตรงกลาง คือ ซอฟต์แวร์ที่ใช้ทดสอบ ในที่นี้คือจาวาคอมไพเลอร์ (javac) โดยผมใช้ source code ของ Linpack.java ที่ได้มาจาก http://www.netlib.org/benchmark/linpackjava/ เพื่อใช้ให้จาวาคอมไพเลอร์ทำงาน

ลงมือทดสอบ: เมื่อเรียกคำสั่ง timejavac ที่ command prompt ดังรูปที่ 2

รูป 2 พิมพ์คำสั่ง **timejavac.bat** ที่ **command prompt** และกดปุ่ม **<ENTER>.**

แล้วเราจะได้ผล เช่น ข้างล่างนี้

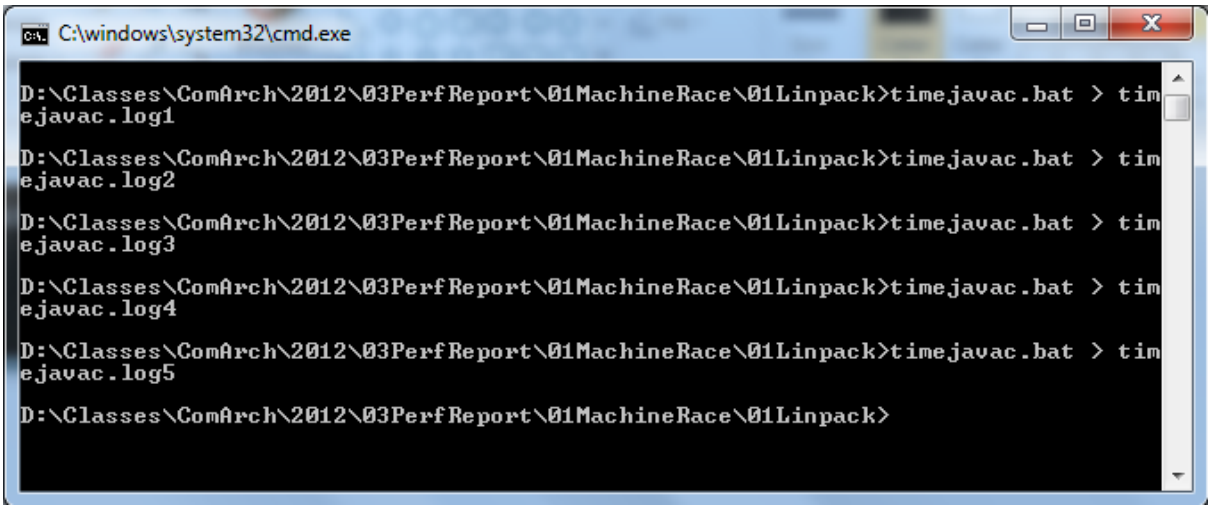
```
D:\Classes\ComArch\2012\03PerfReport\01MachineRace\01Linpack>echo 12:55:13.47
12:55:13.47

D:\Classes\ComArch\2012\03PerfReport\01MachineRace\01Linpack>javac Linpack.java

D:\Classes\ComArch\2012\03PerfReport\01MachineRace\01Linpack>echo 12:55:14.38
12:55:14.38
```

ซึ่งหมายถึง โปรแกรมใช้เวลา ทั้งหมด 0.91 วินาที (จาก 12:55:14.38 - 12:55:13.47 = 0.91) ในการแปลงโค้ดภาษาจาวา Linpack.java ไปเป็นไบค์โค้ด Linpack.class.

เพื่อลดผลจากความแปรปรวนเนื่องจากโปรแกรมนี้ใช้เวลาไม่นาน เราจะทำ 5 ครั้งดังรูปที่ 3. ในรูปที่ 3 แทนที่เราจะคอยอ่านผลจากหน้าจอแต่ละครั้ง เราใช้ direction operator (“>”) ของ DOS Command เพื่อบอกให้เปลี่ยนการแสดงผลไปเก็บไว้ที่ไฟล์ชื่อ timejavac.log1 ถึง timejavac.log5 สำหรับการทำซ้ำที่ 1 ถึง 5 ตามตัวเลขตัวท้ายสุด.



รูป 3 จับเวลา **javac** ทำงาน ทำซ้ำ **5** ครั้ง

รวบรวมผล: รูปที่ 4 เป็นผลที่รวบรวมมาจากไฟล์ผลลัพธ์ทั้ง 5 แล้วเขียนใส่ลงใน MS Excel.

F6

	A	B	C	D	E
1	javac Linpack.java				
2					
3	Repetition	start time	end time	run time	
4	1	12:55:13.47	12:55:14.38	0.91	s
5	2	12:55:19.54	12:55:20.40	0.86	s
6	3	12:55:27.00	12:55:27.97	0.97	s
7	4	12:55:31.66	12:55:32.55	0.89	s
8	5	12:55:36.16	12:55:37.23	1.07	s
9			average	0.94	s
10					

รูป 4 รวบรวมผลการทดสอบ

การทดสอบนี้ใช้ คอมไพเลอร์ javac 1.7.0_05 ของบริษัท Oracle Corporation.

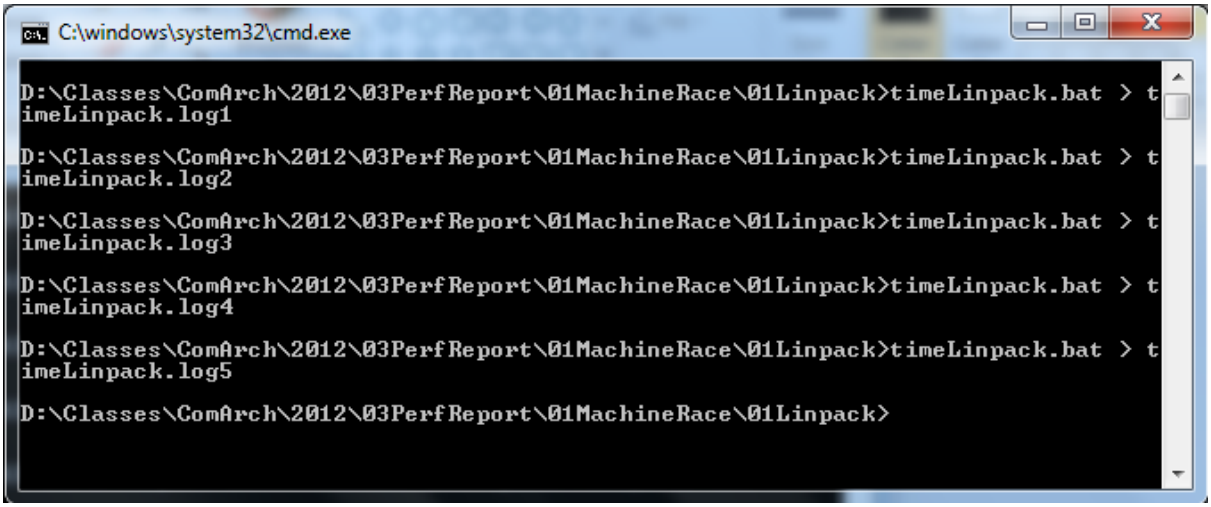
สิ่งที่ควรตรวจสอบ: นอกจากเวลา (ซึ่งเป็นดัชนีวัดที่เราต้องการ) เราควรตรวจสอบด้วยว่า ผลของการทำงานของซอฟต์แวร์ที่รัน เป็นไปอย่างถูกต้อง

<u>หมายเหตุ</u> เพื่อความสะดวกและลดผลจากการเวลาตอบสนองของผู้ใช้ (ซึ่งแปลตามแต่ละบุคคล) การทดสอบความเร็วของคอมพิวเตอร์มักจะเลือกรันซอฟต์แวร์ที่ไม่ต้องเกี่ยวกับการตอบสนองของผู้ใช้

ตัวอย่างข้างต้นเป็นการทดสอบด้วย “javac Linpack.java”, การทดสอบกับซอฟต์แวร์ตัวอื่นๆก็สามารถทำได้ในลักษณะเดียวกัน เช่น

Linpack

ผมใช้ Linpack.class ที่ได้จากการทดสอบแรก แล้วรันด้วย java Linpack ได้แสดงในรูปที่ 5.



รูป 5 วัดประสิทธิภาพด้วย **Linpack**

ผลของการทดสอบด้วย Linpack แสดงดังตารางที่ 1.

Repetition	start time	end time	run time	
1	12:56:24.24	12:56:24.57	0.33	s
2	12:56:28.42	12:56:28.82	0.40	s
3	12:56:32.57	12:56:32.90	0.33	s
4	12:56:36.00	12:56:36.36	0.36	s
5	12:56:41.04	12:56:41.38	0.34	s
		average	0.352	s

ตารางที่ 1 ผลการทดสอบด้วย Linpack

การทดสอบนี้ใช้ Linpack java version (last modified date: Mar 4th, 1997) ที่ดาวน์โหลดจาก http://www.netlib.org/benchmark/linpackjava/ และ คอมไพล์ และรันด้วย javac 1.7.0_05 และ java 1.7.0_05 ตามลำดับ.

<u>หมายเหตุ</u> ผลการทำงานของ Linpack ถ้าดูที่ผลการทำงานของ Linpack เช่น

```
java Linpack

Mflops/s: 2147483.647   Time: 0.0 secs   Norm Res: 1.43   Precision: 2.220446049250313E-16
```

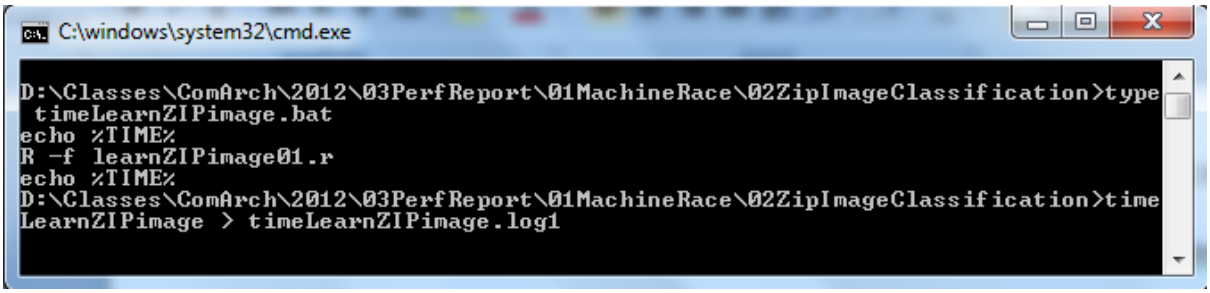
โดย Linpack เองก็เป็นซอฟต์แวร์ทดสอบคอมพิวเตอร์. กล่าวโดยสั้นคือ Linpack จะคำนวณหาคำตอบ x ของระบบสมการเชิงเส้น Ax = b โดยขนาดของเมตริกซ์ A จะเป็น 500 x 500 และ b เป็นเวเตอร์ขนาด 500. วิธีที่ใช้ในการคำนวณจะใช้หลักการของ Gaussian elimination with partial pivoting.

ผลที่แสดงออกมา Mflops/s คือ Millions of floating point operations per second ระบุถึง จำนวน

คำสั่งแบบจุดลอยตัว[1] (ในหลักล้านคำสั่ง) ต่อวินาที; Time เวลาที่ใช้ในการคำนวณ Ax = b; Norm Res ใช้สำหรับตรวจสอบความถูกต้องของผลลัพธ์จากการคำนวณ (ถ้ามีค่ามากกว่า 1 มากๆ ผลการคำนวณอาจจะผิด); และ Precision แสดงค่าความละเอียดเที่ยงตรงของคอมพิวเตอร์. รายละเอียดของ Linpack และผลที่รายงานออกมา ศึกษาเพิ่มเติมได้จาก เวป http://www.netlib.org/benchmark/linpackjava/.

Neural Network Classifier

โปรแกรมนี้เรียนรู้เพื่อจะระบุตัวเลขจากรูปของตัวเลข โดยใช้วิธีโครงข่ายประสาทเทียม. โปรแกรมนี้ learnZIPimage01.r รันบน R statistics[2] โดยที่ Command Prompt จะเรียก R -f learnZIPimage01.r ดังแสดงในเนื้อหาของไฟล์ timeLearnZIPimage.bat ที่พิมพ์ออกมาโดยคำสั่ง type (เนื้อหาของ batch file แสดงบรรทัดที่ 3 ถึง 5 ของรูปที่ 6). บรรทัดที่ 6 – 7 ของรูปที่ 6 เรียก batch file และเก็บการแสดงผลไว้ที่ไฟล์ timeLearnZIPimage.log1.



รูป 6 ทดสอบด้วย **neural network classifier**

เราใช้ ZIP Image dataset (จาก http://www-stat.stanford.edu/~tibs/ElemStatLearn/) ประกอบกับโปรแกรม learnZIPimage01.r. ไฟล์ ZIP image data และไฟล์ประกอบอื่นๆ สามารถดาวน์โหลดได้จาก <????? somewhere on e-learning>.

ผลของการทดสอบด้วย Neural Network Classifier (learnZIPimage01.r) แสดงดังตารางที่ 2.

Repetition	start time	end time	run time	
1	14:06:14.77	14:18:37.16	742.39	s
2	14:27:58.94	14:40:19.49	740.55	s
3	18:21:55.04	18:34:29.76	754.72	s
		average	745.89	s

ตารางที่ 2 ผลการทดสอบด้วย Neural Network Classifier

---

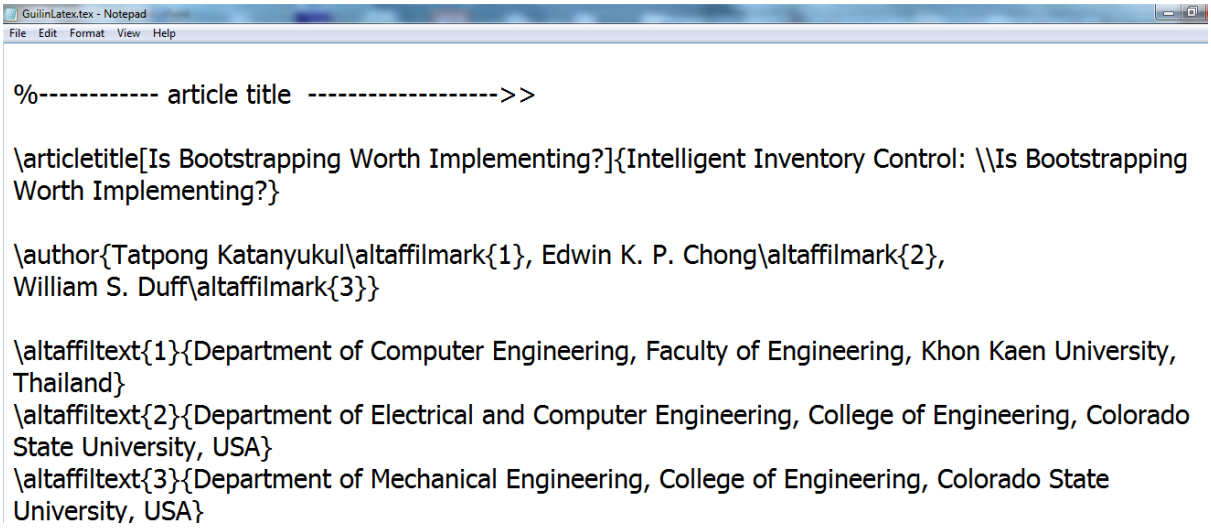
[1] คำสั่งแบบจุดลอยตัว คือ คำสั่งที่ใช้คำนวณแบบทศนิยม, แนะนำอ่านเพิ่มเติมเรื่อง Floating Point Data Representation สำหรับรายละเอียด

[2] สำหรับคนที่ไม่รู้จัก R Statistics ให้คิดถึง R statistics ว่าเป็นคล้ายๆ Java โดยที่เราสามารถเขียนสคริปต์ ในภาษา R แล้วรันได้โดยเรียก **R –f ชื่อไฟล์** คล้ายกับที่เรารันโปรแกรมจาวาด้วย **java ชื่อไฟล์**.

การทดสอบนี้ใช้ชุดข้อมูล ZIP Image dataset จาก http://www-stat.stanford.edu/~tibs/ElemStatLearn/ ภายใต้หัวข้อ ZIP image dataset, learnZIPImage01.r ซึ่งเป็น R script file ที่เขียนขึ้นเอง, และรันด้วย R statistics Version 2.11.1 ซึ่งดาวน์โหลดได้จาก http://www.r-project.org/.

Latex

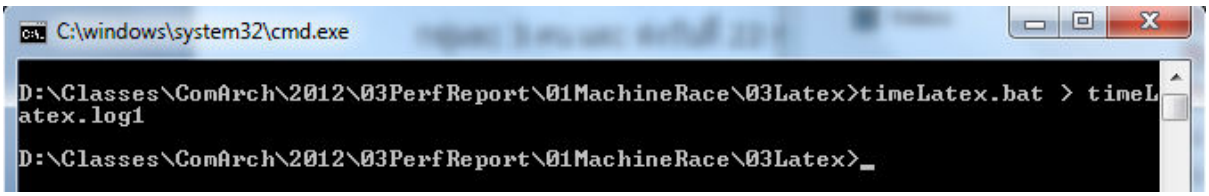
โปรแกรม Latex ใช้สำหรับสร้างเอกสาร เช่น dvi, ps, หรือ pdf. ในการทดสอบนี้เราจะใช้ Latex สร้างบทความจากเนื้อหาที่เขียนใน text file ธรรมดาดังแสดงในรูปที่ 7 ซึ่งผลลัพธ์จะได้เป็นเอกสารที่มีการจัดรูปแบบแล้วดังแสดงในรูปที่ 8.



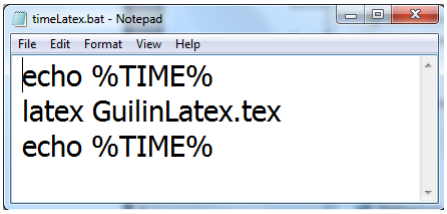
รูป 7 **Latex main input file เป็น plain text**

รูป 8 ผลลัพธ์ของ **Latex** จะเป็นเอกสารที่จัดรูปแบบแล้ว เช่น **pdf, ps,** หรือ **dvi** แบบที่แสดงนี้

การทดสอบนี้ คำสั่งที่ใช้รัน latex คือ **`latex GuilinLatex.tex`** โดย GuilinLatex.tex เป็นไฟล์เนื้อหาในรูปของ plain text file และเรียกผ่าน batch file timeLatex.bat ดังแสดงในรูปที่ 9 และเนื้อหาของ timeLatex.bat แสดงในรูปที่ 10.



รูป 9 เรียก **batch file** เพื่อจับเวลารันเลเท็กซ์



รูป 10 เนื้อหาของ **timeLatex.bat**

ผลการทดสอบด้วย Latex แสดงดังตาราง

Repetition	start time	end time	run time	
1	16:36:42.32	16:36:43.28	0.96	s
2	16:45:29.02	16:45:30.08	1.06	s
3	16:45:45.95	16:45:46.88	0.93	s
		average	0.983333	s

ตารางที่ 3 ผลการทดสอบด้วยเลเท็กซ์

การทดสอบนี้ ใช้โปรแกรมเลเท็กซ์ MiKTeX-pdfTeX 2.8.3512 (1.40.10) หรือ MiKTeX 2.8 ที่สามารถดาวน์โหลดได้จาก http://miktex.org/.

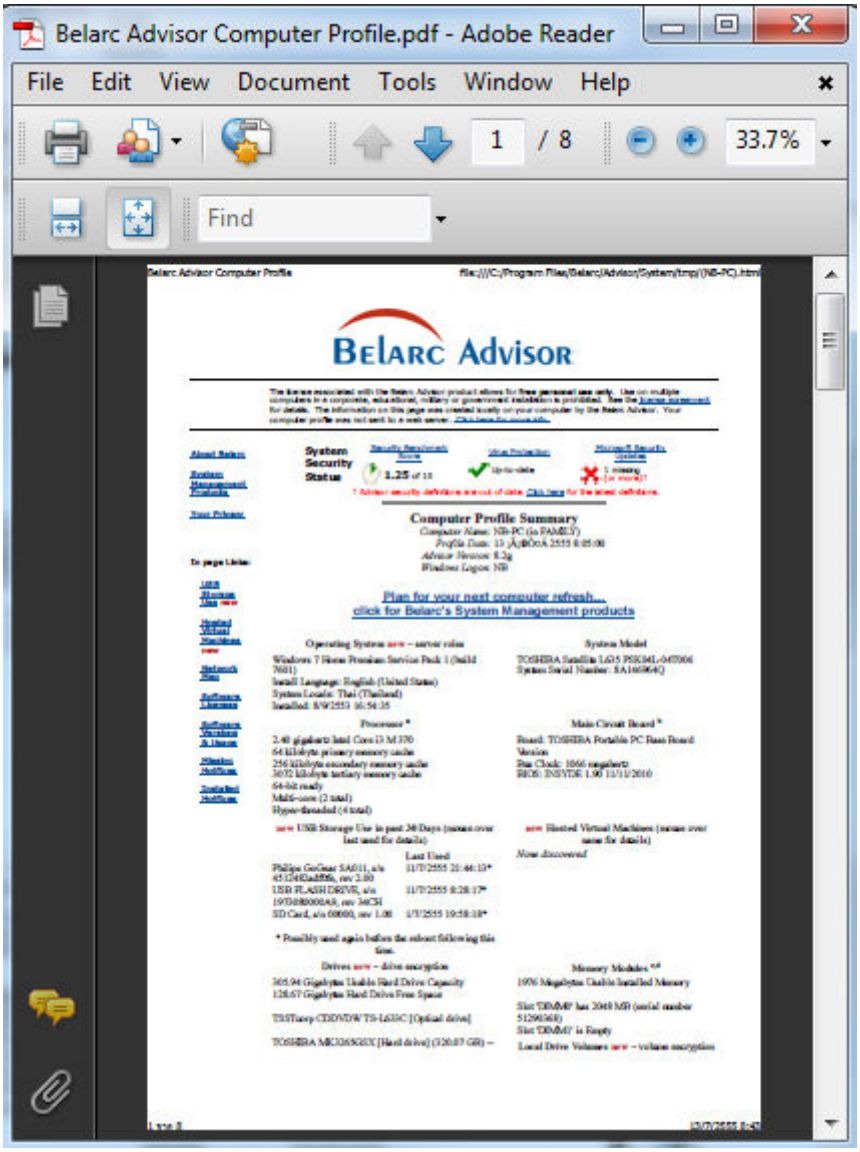
รายละเอียดของเครื่องคอมพิวเตอร์ที่ถูกทดสอบ

System Description

Hardware	
Hardware Vendor	Toshiba
Model	Satellite L635 PSK04L-04T006
CPU	Intel Core i3 M370
CPU clock speed	2.7 GHz
L1 Cache	64KB (on chip)
L2 Cache	256KB (on chip)
L3 Cache	3072KB (on chip)
Bus Clock	1066 MHz
Memory	2048 MB (DDR3 SDRAM)
Main Data Storage	HDD 320.07GB (Toshiba MK3265GSX)
Software	
OS	Windows 7 Home Premium, SP 1
File system type	NTFS
Disk Usage	2 partitions: 143GB primary (62.5 GB free space) and 141GB secondary (57.7GB free space)

ตารางที่ 4 รายละเอียดเครื่องที่ถูกทดสอบ

รายละเอียดของระบบดังข้างต้น สามารถใช้โปรแกรม Belarc Advisor[3] ช่วยรวบรวมให้ได้ (ตัวอย่างของรายงานที่ Belarc Advisor เตรียมให้แสดงดังรูปที่ 11.



รูป 11 รายงานรายละเอียดของเครื่องที่ Belarc Advisor เตรียมให้

สรุปการทดสอบ

จากการทดสอบประสิทธิภาพคอมพิวเตอร์ของทั้ง 4 โปรแกรมเวลาเฉลี่ยคือ 187.04 s (ย่อ ave. ตารางที่ 5) ซึ่งจะเป็นว่าเวลานี้ได้รับอิทธิพลมาจากโปรแกรมที่ใช้เวลานานที่สุด (neural network classifier) ดังนั้นค่าเฉลี่ยแบบนี้จึงไม่เหมาะสม (benchmark ตัวอื่นจะไม่มีผลต่อสรุปเลย)

---

3 ดาวน์โหลดจาก http://download.cnet.com/Belarc-Advisor/3000-2094_4-10007277.html.

การใช้เฉลี่ยตามน้ำหนัก (weighted average, ย่อ w. ave. ตารางที่ 5) ก็ยังปรับค่าน้ำหนักให้เหมาะสมได้ยาก เนื่องจาก benchmark ที่ 3 ใช้เวลานานกว่า benchmark อื่นๆ 700 ถึงกว่า 2,000 เท่าตัว

คอลัมน์ขวาสุดของตารางที่ 5 (normalized) แสดงค่าทำมาตรฐาน (normalized values) ของเวลาเฉลี่ยแต่ละโปรแกรม โดย ค่าทำมาตรฐาน norm = (เวลา – min)/(max – min) เช่น ค่าทำมาตรฐานของ javac จะเป็น (0.94 – 0.86)/(1.07 – 0.86) = 0.381.

ค่าทำมาตรฐานของเวลาเฉลี่ยจาก benchmarks ทั้งสี่ตัวอยู่ในระดับค่าเดียวกัน (ค่าทำมาตรฐาน จะอยู่ระหว่าง 0 ถึง 1) และเราสามารถทำเฉลี่ยตามน้ำหนักเพื่อให้น้ำหนักกับโปรแกรมที่ใช้บ่อยกว่าได้. ค่าเฉลี่ยตามน้ำหนักของค่าทำมาตรฐาน สำหรับ Computer 1 จาก benchmarks ทั้งสี่คือ 0.388 ดังแสดงที่ช่องสุดท้ายของตารางที่ 5.

Computer 1: Satellite L635					
benchmark	average (unit: s)	weight	min (unit: s)	max (unit: s)	normalized
program 1 javac	0.94	0.1	0.86	1.07	0.381
program 2 Linpack	0.35	0.1	0.33	0.40	0.314
program 3 learnZIPimage01	745.89	0.3	740.55	754.72	0.377
program 4 latex	0.98	0.5	0.93	1.06	0.410
	ave. (unit: s)	w.ave. (unit: s)			w.ave.norm.
	187.04	224.39			0.388

ตารางที่ 5 สรุปผลการทดสอบ

เปรียบเทียบ

เนื่องจากผมทดสอบเครื่องเดียวผมจะสมมติตัวเลขของอีกหนึ่งเครื่องเพื่อเปรียบเทียบ สมมติว่าอีกเครื่องทดสอบแล้วสรุปผลได้ตามตารางข้างล่างนี้

Computer 2				
	average (unit: s)	min	max	normalized
program 1	1.02	0.12	1.39	0.708
program 2	3.45	0.24	4.57	0.742
program 3	603.47	315.32	1033.00	0.401
program 4	4.66	3.43	8.45	0.244

ตารางที่ 6 แสดงการประสิทธิภาพของ Computer 1 และ Computer 2 โดยสรุป ค่าเฉลี่ยเวลา, ค่าเฉลี่ยเวลาตามน้ำหนัก, ค่าเฉลี่ยเวลาทำมาตรฐาน, และ ค่าเฉลี่ยเวลาทำมาตรฐานตามน้ำหนักของ Computer 1 และ Computer 2.

เมื่อเปรียบเทียบกันพบว่า ค่าเฉลี่ยเวลา และ ค่าเฉลี่ยเวลาตามน้ำหนัก บอกว่า Computer 2 เร็วกว่า Computer 1.

แต่ ค่าเฉลี่ยเวลาทำมาตรฐาน บอกว่า Computer 1 เร็วกว่า และ ค่าเฉลี่ยเวลาทำมาตรฐานตามน้ำหนัก บอกว่าทั้งสองเครื่องเร็วเท่าๆกัน (ตารางที่ 7 แสดงดัชนีที่ใช้วัดและความหมาย).

เนื่องจากการทำค่ามาตรฐาน มีความเหมาะสมตามเหตุผลที่กล่าวข้างต้น และ เราสามารถปรับค่าน้ำหนักเพื่อสะท้อนปริมาณการใช้งาน การศึกษานี้สรุปว่า Computer 1 และ Computer 2 เร็วเท่ากัน ตามดัชนี ค่าเฉลี่ยเวลาทำมาตรฐานตามน้ำหนัก.

Benchmark	Weight	Computer 1			Computer 2	
		average	normalized		average	normalized
		(unit: s)			(unit: s)	
program 1	0.1	0.94	0.381		1.02	0.708
program 2	0.1	0.35	0.314		3.45	0.742
program 3	0.3	745.89	0.377		603.47	0.401
program 4	0.5	0.98	0.410		4.66	0.244
	Average	187.040	0.371		153.150	0.524
	Weighted Average	224.386	0.388		183.818	0.388

ตารางที่ 6 ประสิทธิภาพคอมพิวเตอร์

ตารางที่ 7 สรุปเปรียบเทียบประสิทธิภาพระหว่างคอมพิวเตอร์ Computer 1 และ Computer 2

Measure	Perf 1/ Perf 2	Perf 2/ Perf 1	Read
average time	0.818809	1.221286	Computer 2 is 1.22 times faster than Computer 1.
weighted average time	0.819204	1.220697	Computer 2 is 1.22 times faster than Computer 1.
average normalized time	1.414063	0.707182	Computer 1 is 1.41 times faster than Computer 2.
weighted average normalized time	1.000002	0.999998	Computer 1 and 2 are the same as fast.