최신 마이크로프로세서상에서 LU-SGS 코드의 국소화 작성

Localized Composition of LU-SGS Code on Latest Microprocessors

*최정열[1)]
Choi, Jeong-Yeol

An approach of composing a performance optimized computational code is suggested for latest microprocessors. The approach named as localization is a concept of minimizing the access to system's main memory and maximizing the utilization of second level cache that is common to all the latest computer system. The localized compositions of LU-SGS scheme for fluid dynamics were made in three different levels and tested on three different microprocessor architectures most widely used in these days. The test results of localization concept showed a remarkable performance, that is the showing gain up to 4.5 times faster solution than the baseline algorithm 450% for producing an exactly the same solution.

1. 서 론

개인용 컴퓨터가 등장한 이후 지난 20년 동안 마이크로프로세서의 성능은 Moore 의 법칙에 따라 매 18 개월마다 2 배씩 증가하는 기하 급수적인 발달을 계속하고 있다.[1] 이러한 프로세서의 성능 향상은 작동 속도(Clock Speed)의 증가는 물론, 회로 집적도의 향상에 따른 프로세서 구조의 발달에 힘입은 바 크다. 현재 생산되고 있는 프로세서의 회로 폭은 100nm 에 근접하고 있으며, 물질의 기본 단위인 원자의 크기와 비교할 때, 조만간 기술 발전의 한계에 도달할 것으로 보인다. 그러나 가공 기술 및 회로 구성 기술 발달에 힘입어 현재까지는 그 동안의 발전 추세가 계속되고 있으며, 최신의 프로세서에는, 이전에 슈퍼컴퓨터에서 이용되었던 파이프라이닝, 벡터처리 및 병렬처리 회로 등이 슈퍼스칼라 구조의 단일 프로세서에서 구현되고 있는 상황이다.[2-4] 실제로 현재 일반적으로 판매되고 있는 x86 계열 프로세서의 경우 1GHz 의 속도에 이르고 있으며, 이는 슈퍼컴퓨터에서나 가능했던 GFlops의 속도가 단일 프로세서에 의해 구현 가능함을 보여주는 예이다.

이러한 프로세서의 고속화 경향에 따라, 계산에 소요되는 시간도 프로세서 내부에서 계산에 소요되는 시간보다 프로세서에 비해 상대적으로 느린 외부 기억장치, 즉, 주기억장치 (RAM)와의 사이에서 데이터를 찾고 주고받는데 많은 시간이 소요되는 경향을 보여준다. 이를 위하여 문제 해결의 근본적인 방법인 고속의 대용량 주기억장치를 개발하는 방법 이외에, 프로세서와 동일한 속도로 작동하는 고속의 임시 저장 장치(캐시)를 256 KB 이상 프로세서 내부에 포함하는 형태로 개발되고 있으며, 프로세서 외부에는 주기억장치보다는 고속인 대용량의 3차 캐시를 이용하는 방법까지 고려되고 있다.

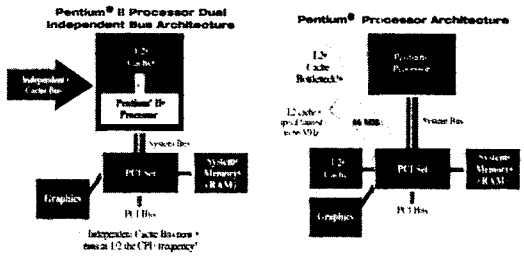

Fig. 2 Pentium 프로세서와 Pentium II 프로세서의 데이터버스 구조

1) 부산대학교 항공우주공학과 (609-735 부산시 금정구 장전동 산 30 Tel : 051-510-2373)

Fig. 1 은 펜티엄 프로세서와 펜티엄 II 프로세서의 구조를 보여주는 그림이다. 이 그림에서 수 년전의 펜티엄 프로세서에서는 캐시와 주 기억장치 모두 동일한 데이터 버스를 공유하여 병목 현상이 발생하였고, 캐시의 작동 속도에 한계가 잇었으나, 펜티엄 II 프로세서는 캐시를 위한 전용 버스 (BSB, Back-side Bus)를 주기억장치 등의 외부 장치를 위한 버스 (FSB, Front-Side Bus)와 분리 함으로써, 병목 현상을 없애고 캐시의 작동속도를 프로세서 작동 속도의 절반까지 향상시킬수 있었다. 현재 이용되는 Pentium III 프로세서에서는 캐시가 프로세서와 통합됨으로써, 프로세서와 동일한 속도로 작동하게 되었으며, 이러한 프로세서의 AMD 의 Athlon 프로세서나 Compaq 의 Alpha 프로세서 등, 최근에 주로 이용되고 있는 다른 종류의 프로세서에서도 동일하게 채용되고 있다.

이러한 프로세서 구조의 변화는 많은 기억 용량과 계산 시간을 소요하며, 무수히 많이 주기억장치에 접근이 이루어져야 하는 전산 유체 해석에 있어서, 코드의 벡터화나 병렬화와는 또 다른 프로그래밍 패러다임을 요구하고 있으며, 이는 "국소화 (localization)" 라는 말로 정리할 수 있다. 즉, 프로세서의 1/5-1/10 정도의 작동 속도를 가지는 주기억장치에 대한 접근을 최소화하고 주기억장치에서 가져온 값의 활용도를 높여 캐시의 적중률(hit ratio)을 높이는 방법이다. 그러나 문제는 프로그램의 수행 중 어떤 변수가 주기억장치에 있는지, 또는 캐시 상에 있는지 일반적으로는 알 수 없다는 것이며, 프로그래머는 캐시의 용량을 고려하여 적절히 코드를 작성하는 방법밖에 없다는 것이다.

따라서 본 연구에서는, 주기억장치에서 가져온 변수의 활용도를 높여 캐시의 적중률을 높일 수 있는 국소화의 개념을 제시하고, 여러 가지의 유동 해석 방법 가운데 국소화를 가장 쉬운 방법으로 구현할 수 있을 것으로 사료되는 LU-SGS[6] 기법에 적용하여, 일반적으로 많이 이용되는 몇 가지 프로세서의 종류와 운용환경에 대하여 국소화에 따른 성능 향상을 살펴보았다.

2. 프로그램 국소화의 단계

2.1 LU-SGS 해법

본 연구에서 시범적으로 고려한 유동 해석 코드는 LU-SGS[6] 기법을 이용하는 비 점성 압축성 유동의 해석 코드이다. 이 해석 코드를 택한 이유는 가장 기본적인 형태의 유동 해석 코드로 여겨지며, 국소화의 개념을 가장 쉽게 적용시킬 수 있을 것으로 여겨졌기 때문이다.

압축성 유동의 지배방정식을 일반 좌표 계에서 벡터형으로 다음과 같이 적을 수 있으며,

$$\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}=0 \qquad (1)$$

여기에, 수치 근사를 하고 LU-SGS을 적용하여 아래와 같이 2단계의 과정으로 풀 수 있다.

$$RHS_{i,j} = F_{i-1/2}-F_{i+1/2}+G_{j-1/2}-G_{j+1/2} \qquad (2)$$

$$D^{n}_{ij}\Delta Q^{*}_{ij} = RHS_{i,j}+A^{n,+}_{i-1}\Delta Q^{*}_{i-1}+B^{n,+}_{j-1}\Delta Q^{*}_{j-1} \qquad (3)$$

$$D^{n}_{ij}\Delta Q^{n}_{ij} = D^{n}_{ij}\Delta Q^{*}_{ij}-A^{n,-}_{i+1}\Delta Q^{n}_{i+1}-B^{n,-}_{j+1}\Delta Q^{n}_{j+1} \qquad (4)$$

그러나 위의 (2)-(4)식을 구현하는 알고리즘은 국소화의 진행정도에 따라 다음과 같이 몇 가지의 방법을 생각하여 볼 수 있다

2.2 국소화 알고리즘 Level 0

위의 식 (2)와 (3)를 계산하여 해를 구하는 기본적인 방법을 Ax=B 형태의 연립방정식의 해를 구하는 과정으로부터 정리하여 생각하여 보면 다음과 같이 4 단계로 정리할 수 있다.

첫째는 보존변수 또는 원시 변수로부터 (2)식의 잔차 벡터를 구하는 단계이다. 여기서 한번 계산된 플럭스 벡터는 다음번 계산 격자에서 다시 이용되기 때문에 한번 계산한 플럭스를 필요한 모든 잔차 벡터에 더해주는 과정이 필요하며, 전체 잔차벡터에 더해 주기 위해 각 방향으로 두 번의 루프를 통하여 계산한다.

A) Compose RHS vector

a1) Do i=1,imax, j=1,jmax

$$F_{i+1/2} \rightarrow RHS_{i,j}\ ,\ RHS_{i+1,j}$$

a2) Do i=1,imax, j=1,jmax

$$G_{j+1/2} \rightarrow RHS_{i,j}\ ,\ RHS_{i,j+1}$$

다음으로 Lower-sweep 단계의 루프에서는 대각 행렬 및 자코비안 행렬 및 변량 벡터와의 곱을 구하여야 하며 이는 벡터화된 간단한 식으로 계산할 수 있다.[7-8]

B) Lower-sweep

Do i=1,imax, j=1,jmax

$$D^{n}_{i,j}\ ,\ A^{n,+}_{i-1}\Delta Q^{*}_{i-1}\ ,\ B^{n,+}_{j-1}\Delta Q^{*}_{j-1} \rightarrow \Delta Q^{*}_{i,j}$$

Upper-sweep 단계의 Lower-sweep 단계와 동일 한 방법으로 역 방향의 루프를 통하여 계산할 수 있다.

C) Upper-sweep

Do i=imax,1, j=jmax,1

$$D^{n}_{i,j}\ ,\ A^{n,-}_{i+1}\Delta Q^{n}_{i+1}\ ,\ B^{n,-}_{j+1}\Delta Q^{n}_{j+1} \rightarrow \Delta Q^{n}_{i,j}$$

마지막 단계로는 계산된 변량으로부터 보존 변수의 새로운 값을 구하고 원시변수등의 필요한 값을 계산하는 단계이다.

D) Solution Update

Do i=1,imax, j=1,jmax

$$\Delta Q^{n}_{i,j} \rightarrow Q^{n+1}_{i,j}\ ,\ [Primitive\ Variables]^{n+1}$$

이상과 같이 기본적인 LU-SGS 알고리즘은 총 5개의 루프로 구성되어 있으며, 다음에 소개되는 국소화 과정을 거쳐 보다 적은 수의 루프로 간략화 될 수 있다.

2.3 국소화 알고리즘 Level 1

주기억장치에서 불러들인 변수의 활용도를 높이는 국소화 방법의 첫 단계로서는, 동일한 연산 횟수와 결과를 유지하면서 A) 단계의 루프를 다음과 같이 줄이는 것을 생각하여 볼 수 있다. 즉, 각 방향으로 플럭스 벡터를 구해서 필요한 잔차벡터에 더해주는 2번의 루프를, 각 격자에서 잔차 벡터를 구하기 위해 필요한 플럭스를 구하고 계산된 플럭스는 다음 격자에서 사용을 위해 임시 기억장소에 저장하는 방법이다. 이때 i 방향의 루프가 내부 루프 일 경우, i 방향 플럭스의 저장을 위해서는 4개의 실수 값이 필요하지만, j 방향 플럭스의 저장을 위해서는 $imax\times 4$개의 실수 값이 필요하게 된다. 이 경우 잔차 벡터를 구하기 위해 필요한 원시 변수를 주 기억 장치로부터 한번만 불러되는 효과를 가져오며, 이후 필요한 임시 기억 장소의 크기는 캐쉬에 포함될 수 있을 만큼 충분히 작다.

A) Compose RHS vector

Do i=1,imax, j=1,jmax

$$F_{i+1/2},\ G_{j+1/2} \rightarrow RHS_{i,j}$$

Replace Flux vectors for later use.

2.4 국소화 알고리즘 Level 2

국소화의 두 번째 단계로서는 RHS 벡터의 계산 과정을 Lower-sweep 과 통합하는 것이다. 즉 임의의 격자점에서 잔차 벡터를 구한 후 바로 lowe-sweep을 진행하는 것으로써, 이렇게 하면 $imax\times jmax\times 4$ 개의 실수로 구성된 RHS 벡터를 광역변수로서 선언할 필요가 없어지며, 임시 상수로서만 4개 실수의 RHS 벡터만이 필요하다. 이는 큰 광역 변수가 사라지므로 기억 용량의 측면에서도 상당한(약 20-30%) 절감 효과를 가지며, 주 기억 장치에 접근할 필요성도 따라서 없어지게 된다. Level 1과 Level 2의 알고리즘은 변수의 치환 과정만 추가적으로 필요하며 수학적으로는 Level 0 와 완벽히 동일하여, 수학적으로 완전히 동일한 결과를 얻는다.

A,B) Compose RHS vector & Lower Sweep

Do i=1,imax, j=1,jmax

$$F_{i+1/2},\ G_{j+1/2} \rightarrow RHS_{i,j},\ RHS_{i+1,j},\ RHS_{i,j+1}$$

Replace Flux vectors for later use.

$$D^{n}_{i,j}\ ,\ A^{n,+}_{i-1}\Delta Q^{*}_{i-1}\ ,\ B^{n,+}_{j-1}\Delta Q^{*}_{j-1} \rightarrow \Delta Q^{*}_{i,j}$$

2.5 국소화 알고리즘 Level 3

국소화의 마지막 단계로서는 Upper-sweep 과 Solution Update 의 과정을 통합하는 것이다. 임 의의 격자에서 upper-sweep 이 끝나면 필요한 수치적 과정이 종료되므로 해를 재구성할 수 있기 때문이며, upper-sweep 과 동시에 진행되므로 해의 재구성은 역순으로 진행된다.

C,D) Upper-sweep & Solution Update

Do i=imax,1, j=jmax,1

$$D^{n}_{i,j}\ ,\ A^{n,-}_{i+1}\Delta Q^{n}_{i+1}\ ,\ B^{n,-}_{j+1}\Delta Q^{n}_{j+1} \rightarrow \Delta Q^{n}_{i,j}$$

$$\Delta Q^{n}_{i,j} \rightarrow Q^{n+1}_{i,j}\ ,\ [Primitive\ Variables]^{n+1}$$

그러나 이 경우, (4)식은 다음 (5)식과 같이 약간 변형된다. 즉 자코비안 해열이 재구성된 해를 이용하여 계산되며, 이는 수렴 특성에 다소간의 변화를 가져올 개연성이 있다.

$$D_{ij}^{n}\Delta Q_{ij}^{n} = D_{ij}^{n}\Delta Q_{ij}^{*} - A_{i+1}^{n+1,-}\Delta Q_{i+1}^{n} - B_{j+1}^{n+1,-}\Delta Q_{j+1}^{n} \quad (5)$$

이러한 국소화 과정을 거치면 LU-SGS 해법에서 전체 격자에 대한 루프는 두 단계로만 간략히 정리되며, 각 단계의 내부과정이 다소 복잡해지는 문제와 변수를 임시저장소에 치환하여야 하는 문제가 있지만, 동일한 변수가 자주 활용되어 캐시의 적중률을 높일 수 있는 개연성을 가지게 된다.

3. 국소화 알고리즘의 적용 결과

3.1 시험 환경

위의 국소화 알고리즘의 효용성을 살펴보기 위하여 2차원 경사면에서 경사 충격파가 발생하는 문제에 대한 해석을 수행하여 보았다. 이 번 연구에서 문제의 크기, 프로세서의 종류 및 사양, 컴파일러 등의 영향을 살펴보기 위한 테스트를 수행하였으며, 테스트에 이용한 문제의 크기를 Table 1 에 정리하였으며, 이용된 시스템의 하드웨어 및 소프트웨어 사양을 Table 2 에 정리하였다. 해석에 소요된 시간은 디스크 출력이나 화면출력을 배제한 상태에서, 문제에 따라 적절한 정도의 반복횟수에 대한 계산 시간을 내장 함수인 CPU_TIME() 이나 SECNDS() 를 이용하여 얻었다.

3.2 계산 소요시간 비교

여러 시스템에서 각 문제의 해석에 소요된 계산 시간을 Fig. 2-4 에 비교하였다. Level 0-3 의 국소화 이외에 참고를 위하여 AF-ADI 방법을 이용하여 "Small"과 "Medium" 문제에 대해 얻은 결과를 함께 도시하였다. "Large"의 경우 AF-ADI 방법은 계산 용량의 문제로 검토 할 수 없는 시스템이 있어 포함시키지 않았다. 한편, 시스템 별로 최적의 성능을 얻기 위하여 참고 문헌 [9]에서 제시된 컴파일 옵션을 참고하였다.

"Small" 문제는 1.2MB의 작은 기억용량이 캐시의 크기와 큰 차이를 보이지 않는 경우이다. 이 경우, Pentium III 계열에서는 국소화 진행에 따라 소요시간 절약은 명확하지만, 절대적인 차이는 크지 않다. 한편, AF-ADI 방법과 LU-SGS 기법간에 현격한 차이를 보인다. 그러나 Athlon 계열과 Alpha 계열에서는 국소화 단계에 따른 명확한 시간 감소가 눈에 띄며, 최대 50% 이상의 시간 절약이 나타났다. "Medium" 문제는 10.8MB의 기억용량이 필요한 경우로서, 전체 적인 경향은 "Small" 문제와 유사하다.

Table 1 Test Problems

검증문제	격자수	기억용량	반복횟수
Small	100×50	1.2MB	1000
Medium	300×150	10.8MB	100
Large	1000×500	120MB	10

Table 2 Briefs of Test Systems & Compilers[9-11]

Name	System Brief
PIII 650 /Compaq /Intel	Intel Mobile Pentium III 650MHz
	On-Die 32(16+16)KB(L1)+256KB(L2)
	PC100 SDRAM 128MB, 100MHz FSB
	Compaq Presario 801XL, Intel 440MX
	Microsoft Windows Me
PIII 1G /Compaq /intel	Intel Pentium III 1GHz
	On-Die 32(16+16)KB(L1)+256KB(L2)
	PC133 SDRAM 128MB, 100MHz FSB
	ASUS CUSL2-C, Intel 815EP
	Microsoft Windows 2000
Athlon 850 /Compaq /Intel	AMD Athlon 850MHz,
	On-Die 128KB(L1)+256KB(L2)
	PC133 SDRAM 128MB, 266MHz EV6
	AOpen AK73, VIA KT133
	Microsoft Windows 2000
Athlon 900 /Compaq /Intel	AMD Athlon 900MHz
	On-Die 128KB(L1)+256KB(L2)
	PC133 SDRAM 128MB, 266MHz EV6
	ASUS A7V, VIA KT133
	Microsoft Windows 98 2nd Edition
21164 500 /DEC	Digital Alpha 21164 500MHz
	On-Die 16KB(L1)+96KB(L2), Ext. 1MB(L3)
	100 MHz ECC SDRAM 128MB, EV56
	Alpha PC164-LX (Samsung)
	Microsoft Windows NT 4.0 SP6
21264 500 /Compaq	Compaq Alpha 21264 500MHz
	On-Die 256+64 (L1), Ext. 4MB (L2)
	100 MHz ECC SDRAM 512MB, EV6
	Compaq Alphastation XP1000
	Compaq Tru64 UNIX 4.0F
/Compaq	Compaq Visual Fortran 6.5, c:\>df /fast /opt:5
	Compaq Fortran for Tru64UNIX v.5, %f77 /fast /opt:5
/DEC	Digital Visual Fortran 5.0 AXP, c:\>df /fast /opt:5
/Intel	Intel Fortran 5.0, c:\ifl G6 /O3 /Qip /Qunroll [4]

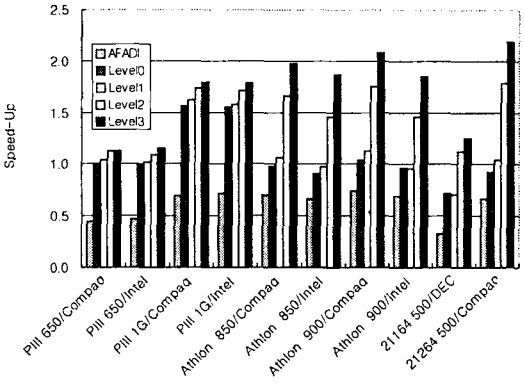

Fig. 2 Speed-Up Ratio for Small Problem

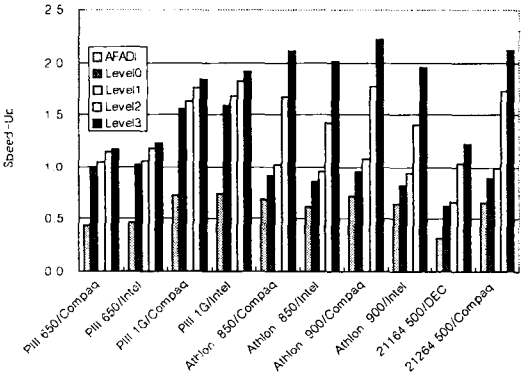

Fig. 3 Speed-Up Ratio for Medium Problem

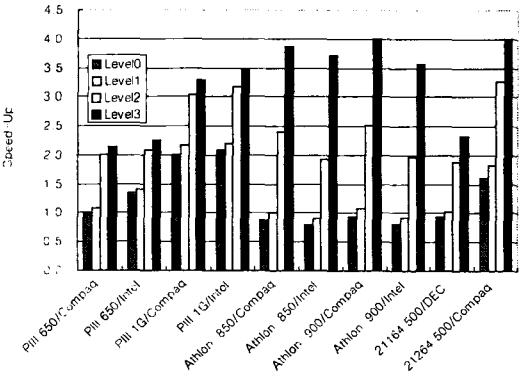

Fig. 4 Speed-Up Ratio for Large Problem

"Large" 문제는 120MB의 매우 큰 기억용량이 필요한 경우로서, 국소화에 따른 성능 향상이 명확히 나타나고 있다. 이 경우 작은 문제들에서는 시간 감소가 크지 않았던 Pentium III 계열에서도 최대 50% 이상의 시간 감소가 나타났으며, Athlon 이나 Alpha 계열에서는 최대 80%에 근접한 시간 감소를 보여 주었다.

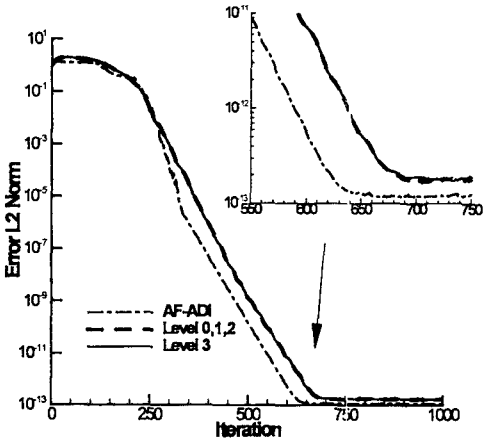

Fig. 5 Comparison of Convergence History

4. 결 론

각 국소화의 단계는, 프로세서의 종류나 운용 환경에 따라 성능 향상의 폭에는 차이가 있으나, 국소화 진행에 따른 일관된 성능 향상을 보여 주었으며, 각 단계 가운데에서는 *RHS* 벡터의 이용을 배제하게 하는 Level 2 의 성능 향상이 가장 크게 나타났다. Level 3 에서는 이전 단계와 수렴 특성 다를 수 있는 개연성이 있으나, 실제의 수렴성을 비교한 경우, 기계 오차에 이르는 동안 이전의 단계와 거의 차이가 없음을 Fig. 5 에서 확인 할 수 있다. 한편 기법에 따른 수렴성 면에서는 AF-ADI가 LU-SGS 보다 우월하나 위 그림에서 나타나는 계산 실 소요 시간을 고려하면 후자가 우월한 것으로 보인다.

한편 운영체제별에 따른 특성은 확인할 수 없었으며, 각 시스템에 따라서는 컴파일러에 따른 성능의 차이가 나타나기도 하였다. 즉, Pentium III 계열에서는 Intel 의 컴파일러가 다소 좋은 결과를 보여 주었으며, Athlon 계열에서는 Compaq 컴파일러가 좋은 결과를 보여 주었다. 한편, Pentium III 계열에서는 국소화에 영향을 비교적 덜 받는 고른 성능을 보여 주었

으며, Athlon 이나 Alpha 21264 프로세서는 국소화가 잘된 경우 최상의 성능을 보여 주었다.

참고문헌

[1] http://www.intel.com/intel/museum/25anniv/hof/moore.htm

[2] Crandall, R.E., "PowerPC G4 for Engineering, Science, and Educatin," http://www.apple.com/powerbook/processor.html

[3] http://www.terms.co.kr/

[4] "P6 Microarchitecture Tuning Guide," http://developer.intel.com/software/asc/documents/04.PDF

[5] L3 cache, http://www.zdwebopedia.com /TERM/L/L3_cache.html

[6] Yoon, S. and Jameson, A., "Lower-Upper Symmetric-Gauss-Seidel Method for the Euler and navier-Stokes Equations," AIAA Journal, Vol.26, No. 9, 1988, pp.1025-1026.

[7] Choi, J.-Y., Jeung, I.-S. and Yoon, Y., "Computational Fluid Dynamics Algorithms for Unsteady Shock-Induced Combustion, Part 1: Validation," *AIAA Journal*, Vol. 38, No. 7, July 2000, pp.1179-1187.[7]

[8] Choi, J.-Y., Jeung, I.-S. and Yoon, Y., "Computational Fluid Dynamics Algorithms for Unsteady Shock-Induced Combustion, Part 2: Comparison," *AIAA Journal*, Vol. 38, No. 7, July 2000, pp.1188-1195.

[9] http://www.kbench.com

[10] http://www.tomsharware.co.kr

[11] http://www.polyhedron.co.uk