Лекция 2. Методы повышения производительности вычислительных систем.

1. Наращивание аппаратных средств. Пример.

Вычислитель	год	Частота, МГц	Увели-чение	Производи-тельность	Увели-чение
EDSAC	1949	0.5	1	100 оп/с ~ 1 Flops	1
HPSuperdome	2001	770	1440	2 GFlops	$2*10^9$
Intel Itanium2	2004	1600	3200	6 GFlops	$6*10^9$
Intel Xeon	2004	3600	7200	4 GFlops	$4*10^9$
Intel Core 2 Duo	2006	3000	6000	2x4=8 GFlops	$8*10^9$
Intel Xeon 64	2007	3730	7460	6x7=42 GFlops	$4.2*10^{10}$
Intel i7-990X, X5690, E7-8800	2011	3460	6920	10x7=70 GFlops	$7*10^{10}$
Nvidia Tesla m2090, 512я	2011	1331	2662	665 GFlops*)	$6.65*10^{11}$
Nvidia Tesla K20X, 2688я	2013	732	1464	1310 GFlops*)	$1.31*10^{12}$
Intel Xeon Phi 5120P, 61я, 244п	2014	1330	2660	1010 GFlops	$1.01*10^{12}$
Nvidia Tesla TCSK40M, 2880я	2015	745	1490	1430 GFlops*)	$1.43*10^{12}$
Nvidia Tesla TCSK80M, 4992я	2016	562	1124	2900 GFlops*)	$2.9*10^{12}$
Nvidia Tesla TCSP100, 3584я	2016	1328	2656	4700 GFlops*)	$4.7*10^{12}$
Intel Xeon Phi KNL 7290X, 36пх2ях4п OmniPath (100 ГБ/с)	2016	1500	3000	3000 GFlops	$3*10^{12}$
NVidia Volta V100, 640 тензорных ядер, 5120 видеоядер (CUDA), NVLINK (300 ГБ/с), PCIE (32 ГБ/с), ОЗУ 32 Гб	2018	1455	2910	7800 GFlops*)	$7.8*10^{12}$
NVidia A100 Tensor Core, 512 тензорных ядер, 8192 видеоядер (CUDA), NVLINK2 (600 ГБ/с), PCIE (64 Гб/с), ОЗУ 40 Гб	2020	1410	2820	9700 GFlops*)	$9.7*10^{12}$
NVidia H100 PCIe Tensor Core, 456 тензорных ядер, 14592 видеоядер (CUDA), NVLINK4 (900 ГБ/с), PCIE (128 Гб/с), ОЗУ 80 Гб	2022	1125 (1755)	2250 (3510)	26000 GFlops (51000 TFlow)	$26*10^{12}$
NVidia H100 SXM Tensor Core, 528 тензорных ядер, 16896 видеоядер (CUDA), NVLINK4 (900 ГБ/с), PCIE (128 Гб/с), ОЗУ 80 Гб	2022	1125 (1755)	2250 (3510)	34000 GFlops (67000 TFlow)	$34*10^{12}$

*) Двойная точность (double precision)

Частота – развитие элем. базы, производительность – развитие архитектуры.

Процессоры Эльбрус.

Советские ЭВМ: серия БЭСМ (большие электронно-счетные машины)

БЭСМ6 (1967) => интегральная БЭСМ6 (1977) => суперкомпьютер Эльбрус (1988).

Процессор Эльбрус-2000 основан на архитектуре ELBRUS (англ. *ExpLicit Basic Resources Utilization Scheduling* — «явное планирование использования основных ресурсов»).

Компьютер Эльбрус-3М при частоте 300 МГц в режиме совместимости с IA-32 обогнал 500 МГц Intel Pentium III в тестах SPEC. При работе в «родных» кодах Эльбруса процессор показал скорость, сравнимую с Pentium 4 2 ГГц. «Эльбрус-2С+» — гибридный (2 ядра «Эльбрус» + кластер из 4-х DSP ядер Мультикор) процессор, 90 нм, 2011.

- «Эльбрус-2СМ» — упрощенный вариант «Эльбрус-2С+» (без кластера DSP ядер), 90 нм, 300 МГц, к 2014.
- «Эльбрус-4С» (другое название «Эльбрус-2S») — 64 Гфлоп, 65 нм, к 2013/2014 г.
- «Эльбрус-1С+» — Экономичный микропроцессор с одним ядром Эльбрус и встроенным графическим ядром, 40 нм, к 2015 г.
- «Эльбрус-8С» — 8 ядер Эльбрус, частота >1 ГГц, производ. > 150 Гфлоп, 28 нм, к 2015 г.
- «Эльбрус-16С» — 1 Тфлоп, ≤28 нм, к 2018 г.

	Эльбрус-2С+	Эльбрус-4С	Эльбрус-8С	Эльбрус-16С
Год выпуска	2011	2014	2015-2018	2018 (план)
Тактовая частота	500 МГц	800 Мгц	1300 МГц	1500 Мгц
Разрядность	хз	32/64 бит	64 бит	64/128 бит
К-во ядер	2	4	8	8/16
Кэш первого уровня	64 Кб	128 Кб	—	—
Кэш второго уровня	1 Мб	8 Мб	4 Мб	4 Мб
Кэш третьего уровня	—	—	16 Мб	16 Мб
Поддержка ОЗУ	DDR2-800	3 x DDR3-1600	4 x DDR3-1600	4 x DDR4-2400
Техпроцесс	90 нм	65 нм	28 нм	28 нм (или 16)
Потребление энергии	25 Вт	45 Вт	75-100 Вт	60-90 Вт

Сравнение производительности

Процессор, № (Имя)	Число ядер	GFlops, с ДТ	Частота, GHz	L3 cache, MB	Техпроцесс, нм	Тип ОЗУ	ОЗУ MAX	Слотов ОЗУ
Core i7 975	4	50	3.3	8	45	DDR3-1066	24	3
Эльбрус-4С (E2S)	4	50	0.8	0	65	DDR3-1600	48	3
2x Xeon x5677	4	104	3.5	12	32	DDR3-1333	288	9
i5-2500K (Sandy Bridge)	4	118	3.3	6	32	DDR3-1333	32	2
Эльбрус-8С (P1)	8	125	1.3	16	28	DDR3-1600	64	8
i7-4960X (Ivy Bridge)	6	244	3.6	16	22	DDR3-1866	64	4
i7-6950X (Broadwell E)	10	247	3.0	26	14	DDR4-3000	128	4
Эльбрус-8С2 (P9)	8	288	1.6	16	28	DDR4-2600	2048	8
i7-5960X (Haswell)	8	350	3.5	20	22	DDR4-2400	128	4
AMD A10-7850K	12	427	3.7	0	28	DDR3-2133	64	4
Эльбрус-16С	16	750	2.0	32	16	DDR4-2600	2048	8
Xeon E7-8890 v4	24	844	2.2	60	14	DDR4-1600	3078	12
Xeon E7-8894 v4	24	921	2.4	60	14	DDR4-1600	3078	12
Эльбрус-32С	32	2000-4000	2.0	≥32	14	DDR4-3200	2048	8
Xeon E5-2699 V5	32	2500	2.1	45	14	DDR4-3200	3078	12
Xeon Phi™ Processor 7210	72	3000	1.5	36	14	DDR4-2400	3078	12

Российские процессоры Байкал-Т1 и Байкал-М.

Процессоры Байкал-Т1 предназначены больше для промышленного сегмента. Процессоры Байкал-Т1 можно использовать для маршрутизаторов, роутеров и другого телекоммуникационного оборудования, для тонких клиентов и офисной техники, для мультимедийных центров, систем ЧПУ.

Процессоры Байкал-М смогут предназначены для рабочих ПК, для промышленной автоматизации и для управления зданиями. Будет работать на 8 ядрах ARMv8-A и иметь на борту до восьми графических ядер ARM Mali-T628. Процессор Байкал-М - система на кристалле, включающая энергоэффективные процессорные ядра с архитектурой ARMv8, графическую подсистему и набор высокоскоростных интерфейсов. Байкал-М может использоваться в качестве доверенного процессора с широкими возможностями защиты данных в ряде устройств B2C и B2B сегментов. Области применения Байкал-М моноблок, автоматизированное рабочее место, графическая рабочая станция; домашний (офисный) медиа-центр; сервер и терминал видеоконференций; микросервер; NAS уровня небольшого предприятия; маршрутизатор / брандмауэр. Широкое распространение архитектуры ARMv8 (AArch64) позволяет использовать огромное количество готового прикладного и системного программного обеспечения. Поддерживаются операционные системы Linux и Android, в том числе на уровне бинарных дистрибутивов и пакетов. Доступны драйверы многочисленных устройств, подключаемых к

шинам PCIe и USB. В состав поставляемого <Байкал Электроникс> комплекта программного обеспечения входит ядро Linux в исходных текстах и скомпилированном виде, а также драйверы для встроенных в Baikal-M контроллеров.
Основные характеристики процессора Байкал-М:
- 8 ядер ARM Cortex-A57 (разрядность 64 бит);
- рабочая частота до 2 ГГц;
- аппаратная поддержка виртуализации и технологии Trust Zone;
- два 64-битных канала DDR3/DDR4-2133 с поддержкой ECC;
- кэш-память - 4 МБ (L2) + 8 МБ (L3);
- 8-ядерный графический сопроцессор Mali-T628;
- видеотракт, обеспечивающий поддержку HDMI, LVDS
- аппаратное декодирование видео;
- встроенный контроллер PCI Express поддерживает 16 линий PCIe Gen. 3;
- 2 контроллера 10-гигабитной сети Ethernet, 2 контроллера гигабитной сети Ethernet;
- контроллеры поддерживают виртуальные сети VLAN и приоритезацию трафика;
- 2 контроллера SATA 6G, обеспечивающих скорость обмена данными до 6 Гбит/с каждый;
- 2 канала USB v.3.0 и 4 канала USB v.2.0;
- энергопотребление - не более 30 Вт.

2. Параллельная обработка данных – эффективный метод повышения производительности.

Два вида параллельной обработки: параллельный и конвеерный.
Пример: сложение векторов **c** = **a** + **b** размерности 10, 5-ти тактное сложение.

T	Последовательный	T	Параллельный, 2 проц.	T	Конвеерный, длина конв.=4
0	a10…a1, b10…b1 -> +	0	a10…a1, b10…b1 -> + +	0	a10…a1, b10…b1 -> ++++
1	a1+b1	1	a1+b1, a2+b2	1	a1+b1
2	a1+b1	2	a1+b1, a2+b2	2	a2+b2, a1+b1
3	a1+b1	3	a1+b1, a2+b2	3	a3+b3, a2+b2, a1+b1
4	a1+b1	4	a1+b1, a2+b2	4	a4+b4, a3+b3, a2+b2, a1+b1
5	a1+b1 = c1	5	a1+b1=c1, a2+b2=c2	5	a4+b4, a3+b3, a2+b2, a1+b1=c1
6	a2+b2	6	a3+b3, a4+b4	6	a5+b5, a4+b4, a3+b3, a2+b2=c2
7	a2+b2	7	a3+b3, a4+b4	7	a6+b6, a5+b5, a4+b4, a3+b3=c3
8	a2+b2	8	a3+b3, a4+b4	8	a7+b7, a6+b6, a5+b5, a4+b4=c4
9	a2+b2	9	a3+b3, a4+b4	9	a8+b8, a7+b7, a6+b6, a5+b5
10	a2+b2=c2	10	a3+b3=c3, a4+b4=c4	10	a8+b8, a7+b7, a6+b6, a5+b5=c5
…	…	…	…	11	a9+b9, a8+b8, a7+b7, a6+b6=c6
46	a10+b10	21	a9+b9, a10+b10	12	a10+b10, a9+b9, a8+b8, a7+b7=c7
47	a10+b10	22	a9+b9, a10+b10	13	a10+b10, a9+b9, a8+b8=c8
48	a10+b10	23	a9+b9, a10+b10	14	a10+b10, a9+b9
49	a10+b10	24	a9+b9, a10+b10	15	a10+b10, a9+b9=c9
50	a10+b10=c10	25	a9+b9=c9, a10+b10=c10	16	a10+b10=c10
	$Q = N_d * N_t$ $P = 1 / (N_t * \tau)$		$Q = N_d * N_t / N_p$ $P = N_p / (N_t * \tau)$		$Q = N_d * N_t / L_k + L_k - 1$ $P = L_k / ((N_t + (1-1/L_k)/N_d) * \tau)$

Q – общее число тактов, N_d – кол-во обработанных данных, N_t – число тактов на операцию, L_k – длина конвеера. Возникли понятия – ***скалярные и векторные команды, скалярные, конвеерные и векторные устройства.*** **Общая формула:** $Q = N_i + N_d * N_t / L_k + L_k - 1$, N_i – кол-во тактов для инициализации векторного устройства. ***Реальная производительность*** любого устройства $P = N_d / T$, T – общее время выполнения задачи. ***Пиковая производительность*** любого устройства $P_{max} = 1/\tau$, τ - длительность такта. $T = Q*\tau \Rightarrow P = N_d / (Q*\tau)$ и P –> P_{max} с ростом N_d у конвеерных и векторных устройств!!!

3. Аппаратные решения, использующие параллелизм.

1) переход от разрядно-последовательной к разрядно-параллельной арифметике
2) переход от разрядно-последовательной к разрядно-параллельной памяти
3) параллельная работа УУ, АЛУ и УВВ за счет введения каналов ВВ (спец. процессоров)
4) совмещение арифм. действий с чтением/записью в память за счет каналов прямого доступа к памяти
5) совмещение вызова команд и данных для них с помощью каналов команд и данных
6) использование одного УУ и многих АЛУ (ФУ) – чистый параллелизм
7) применение конвеерного принципа обработки команд

8) использование многих конвеерных ФУ – параллелизм + конвеерная обработка
9) опережающий просмотр потоков команд и данных, упреждающее выполнение команд
10) применение векторных и матричных ФУ (процессоров)
11) введение векторных команд процессора
Зам.: для увеличения производительности вычислительной системы (ВС) с помощью аппаратного параллелизма необходимо увеличить ***локальность вычислений и локальность используемых данных.***

4. Увеличение интеллектуальности АПС.

Зам.: Реальная производительность зависит от степени поддержки параллелизма в аппаратно-программной среде (АПС) выч. системы. Каждая составляющая АПС вносит вклад в производительность.
С точки зрения управляющей структуры ВС сформировались два класса процессоров:
1) универсальные (центральные) процессоры;
2) специализированные процессоры.
Появились также процессоры, активно использующие параллелизм на уровне машинных команд
1) суперскалярные процессоры – обнаруживают во время выполнения задачи параллелизм в машинном коде и реализуют его – не требуется особая компиляция программ – непредсказуемость результата, динамические возможности;
2) VLIW-процессоры (Very Large Instruction Word) – выполняют длинные команды, состоящие из набора параллельных операций – требуется особая компиляция программ, четко расписывающая параллелизм – предсказуемость результата, ограниченные возможности архитектуры процессора.
Сформировались также два основных класса МВС
1) компьютеры с общей (разделяемой) памятью – **мультипроцессорные системы** – P1->M, P2->M, …
2) компьютеры с распределенной памятью – **мультикомпьютерные системы** – (P1,M1)->KC, (P2,M2)->KC,…
SMP-системы – представители 1) с одинаковыми процессорами (Symmetric Multi Processors).
Появление МВС выявило две проблемы:
а) как достичь их максимальной производительности;
б) как разработать для этого эффективное ПО.
Решение этих проблем сильно зависит от ***системы коммутации процессоров.***
Способы коммутации мультипроцессорных систем:
1) общая шина (недостаток – малое число устройств);
2) разделение памяти на модули;
3) применение матричных и каскадных коммутаторов доступа к модулям памяти
(недостаток матричных – много оборудования - n^2 устройств; у каскадных – задержка доступа из-за длинного пути - кол-во каскадов $log_2 n$ по $n/2$ коммутаторов в каждом; n – число модулей памяти и/или процессоров)
4) другие топологии коммутации: линейка, кольцо, звезда, решетки, тор, двоичный гиперкуб, клика
(зам.: 3) и 4) применяются уже и в мультикомпьютерных ВС)
5) гибридные топологии коммутации
6) кластерное устройство доступа – внутри кластера по общей шине, сами кластеры – шина или коммутация др. типа
Проект NUMA - неоднородный доступ к памяти – единое адресное пространство, локальная и межкластерная шина доступа к распределенным модулям памяти: (P, M, IO, лок. шина, контроллер памяти) -> межкластерная шина
Проблема NUMA – некогерентность кэш-памяти –> ccNUMA (cache coherent NUMA)
Технологии параллельного программирования
1) спец. комментарии в языках высокого уровня, описывающие параллельные части кода (Fortran, C - OpenMP)
2) расширение языков программирования (High Performance Fortran - HPF, mpC)
3) специальные языки параллельного программирования (Occam, НОРМА)
4) библиотеки и интерфейсы, поддерживающие взаимодействие параллельных процессов (PVM, MPI, Linda)
5) параллельные предметные библиотеки (BLAS, Lapack, ScaLapack, Cray Scientific Library, HP Mathematical Library, PETsc)
6) специализированные пакеты и программные комплексы (GAMESS – квантово-химические расчеты, Fluent – газодинамические расчеты, ANSYS – механические расчеты и т.д.)