

Параллельные вычисления (часть 1)

Михаил Георгиевич Курносов

Email: mkurnosov@gmail.com

WWW: <http://www.mkurnosov.net>

Курс «Параллельные и распределённые вычисления»

Школа анализа данных Яндекс (Новосибирск)

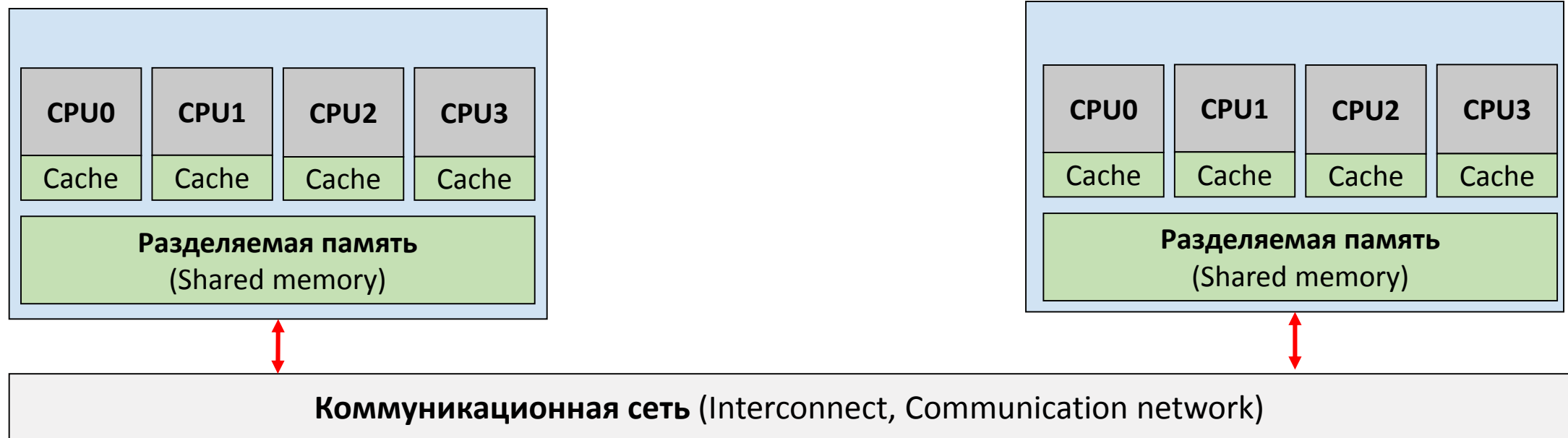
Весенний семестр, 2015

Параллельные вычисления (Parallel Computing)

- **Цели использования многопроцессорных вычислительных систем (ВС):**
 - ❑ решение задачи за меньшее время на нескольких процессорах
 - ❑ решение задачи с большим объёмом входных данных –
использование распределенной памяти нескольких вычислительных узлов
 - ❑ решение задачи с большей вероятностью получения корректного решения
(дублирование вычислений – параллельный пересчет)

Архитектура вычислительных систем с распределенной памятью

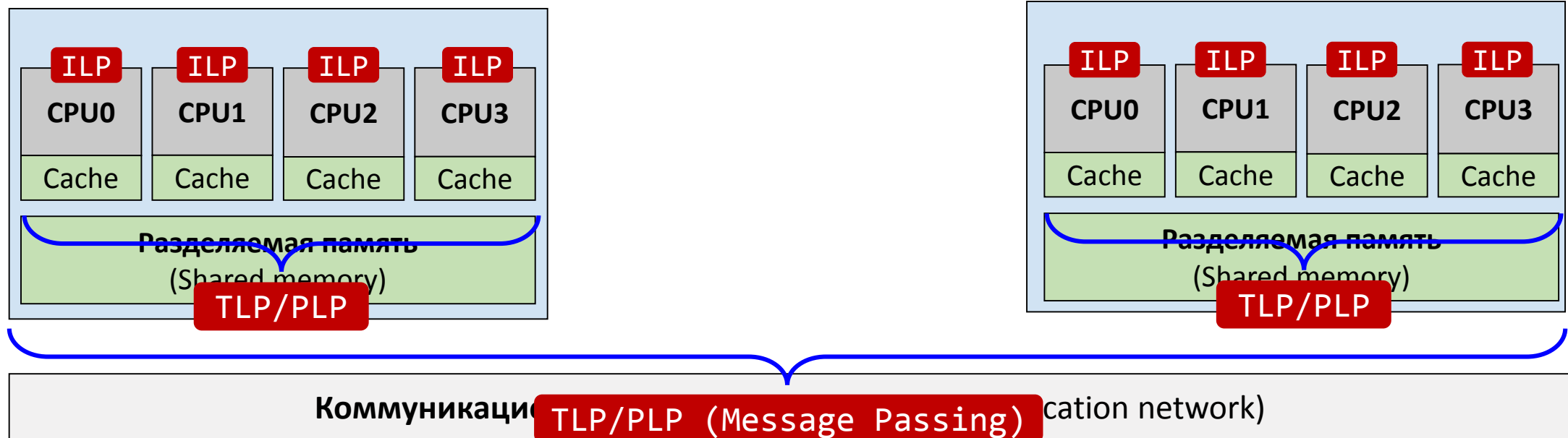
Вычислительные системы с распределенной памятью



- **Вычислительная система с распределенной памятью** (Distributed memory computer system) – совокупность вычислительных узлов, взаимодействие между которыми осуществляется через коммуникационную сеть (InfiniBand, Gigabit Ethernet, Cray Gemeni, Fujitsu Tofu, ...)
- Каждый узел имеет множество процессоров/ядер, взаимодействующих через разделяемую память (Shared memory)
- Процессоры могут быть многоядерными, ядра могут поддерживать одновременную многопоточность (SMT)

- <http://www.top500.org>
- <http://www.green500.org>
- <http://www.graph500.org>
- <http://top50.supercomputers.ru>

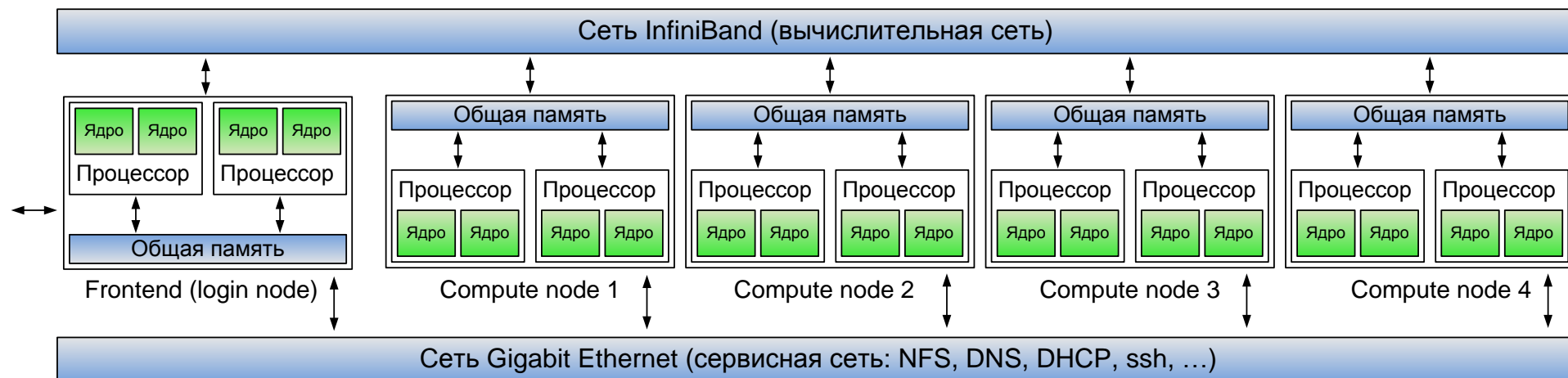
Вычислительные системы с распределенной памятью



- **Вычислительная система с распределенной памятью** (Distributed memory computer system) – совокупность вычислительных узлов, взаимодействие между которыми осуществляется через коммуникационную сеть (InfiniBand, Gigabit Ethernet, Cray Gemeni, Fujitsu Tofu, ...)
- Каждый узел имеет множество процессоров/ядер, взаимодействующих через разделяемую память (Shared memory)
- Процессоры могут быть многоядерными, ядра могут поддерживать одновременную многопоточность (SMT)

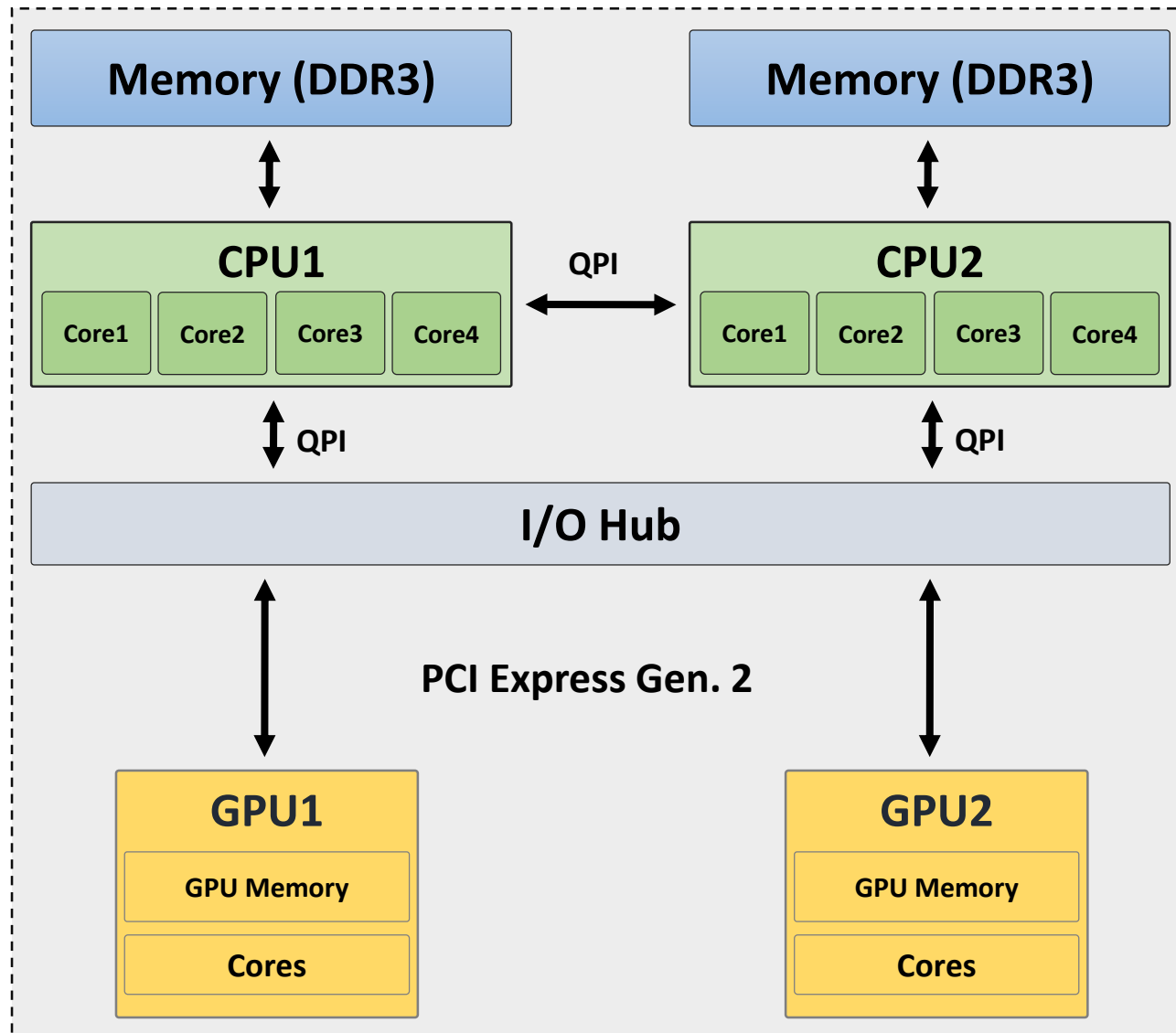
- <http://www.top500.org>
- <http://www.green500.org>
- <http://www.graph500.org>
- <http://top50.supercomputers.ru>

Вычислительные кластеры (Computer cluster)



- **Вычислительные кластеры строятся на базе свободно доступных компонентов**
- Вычислительные узлы: 2/4-процессорные узлы, 1 – 8 GiB оперативной памяти на ядро (поток)
- Коммуникационная сеть (сервисная и для обмена сообщениями)
- Подсистема хранения данных (дисковый массивы, параллельные и сетевые файловые системы)
- Система бесперебойного электропитания
- Система охлаждения
- Программное обеспечение: GNU/Linux (NFS, NIS, DNS, ...), MPI (MPICH2, Open MPI), TORQUE/SLURM

Гибридные вычислительные узлы



- GPU (NVIDIA, AMD)
- Intel Xeon Phi
- FPGA-based accelerators

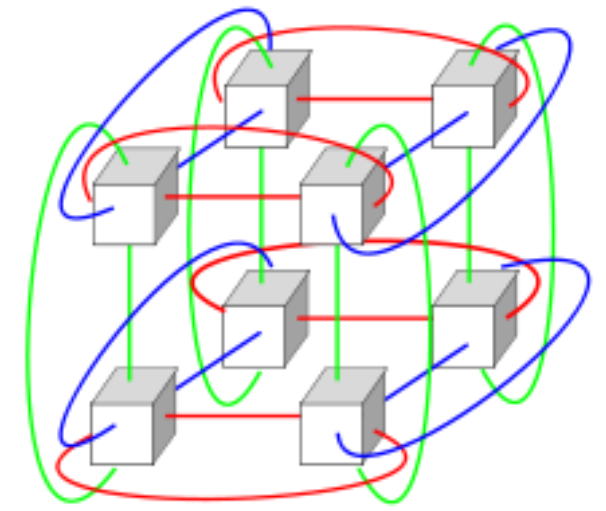
Коммуникационные сети (Interconnect)

- **С фиксированной структурой (статической топологией) – графом связей вычислительных узлов**
- Каждый вычислительный узел имеет сетевой интерфейс (маршрутизатор) с несколькими портами, через который он напрямую соединён с другими узлами
- **С динамической структурой – на базе коммутаторов**
- Каждый вычислительный узел имеет сетевой интерфейс с несколькими портами
- Порты интерфейсов подключены к коммутаторам, через которые происходит взаимодействие узлов



Сети с фиксированной структурой

- **Структура сети (топология)** – это граф, узлы – это машины, а ребра – это линии связи
- **Показатели эффективности структур:**
 - ❑ диаметр (максимальное из кратчайших расстояний)
 - ❑ средний диаметр сети
 - ❑ бисекционная пропускная способность (bisectional bandwidth)
- **Примеры структур**
- ***kD*-тор** (3D, 4D, 5D): Cray Titan (#2) 3D torus, IBM Sequoia 5D torus (#3), Fujitsu 6D torus (#4)
- **Гиперкуб**
- **Решетка, кольцо, графы Кауца, циркулянтные структуры, ...**



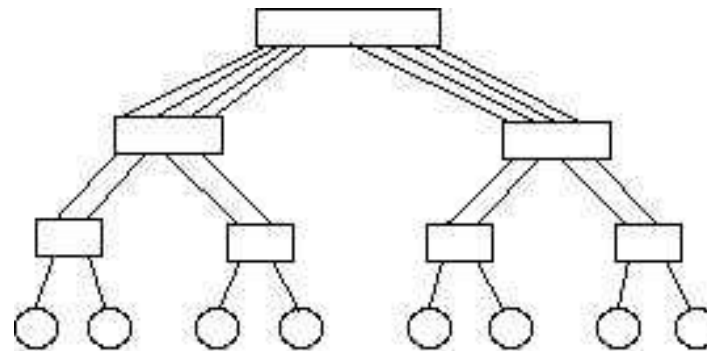
3D-тор

Сети с динамической структурой

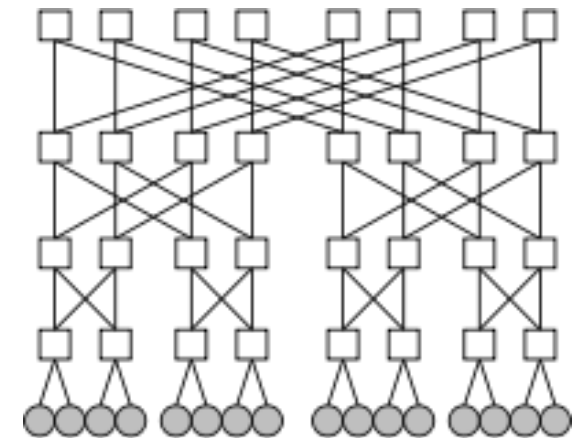
- Вычислительные узлы связаны через активные устройства – сетевые коммутаторы (network switches)
- Коммутатор имеет фиксированное число портов (24, 48, 96, 324)
- Как объединить в сеть тысячи узлов?
- Топология “толстого дерева” (fat tree)

Charles E. Leiserson. **Fat-trees: universal networks for hardware-efficient supercomputing** // IEEE Transactions on Computers, Vol. 34, No. 10, 1985

- **Сетевые технологии**
- Gigabit Ethernet
- 10 Gigabit Ethernet
- InfiniBand DDR/QDR/FDR



Fat tree



Показатели эффективности коммуникационных сетей

- **Пропускная способность (Bandwidth)** – бит/сек. (10 – 54 Gbps – Giga bit per second)
- **Латентность (Latency)** – сек. (10 us – 100 ns)
- **Бисекционная пропускная способность (Bisectional bandwidth)**

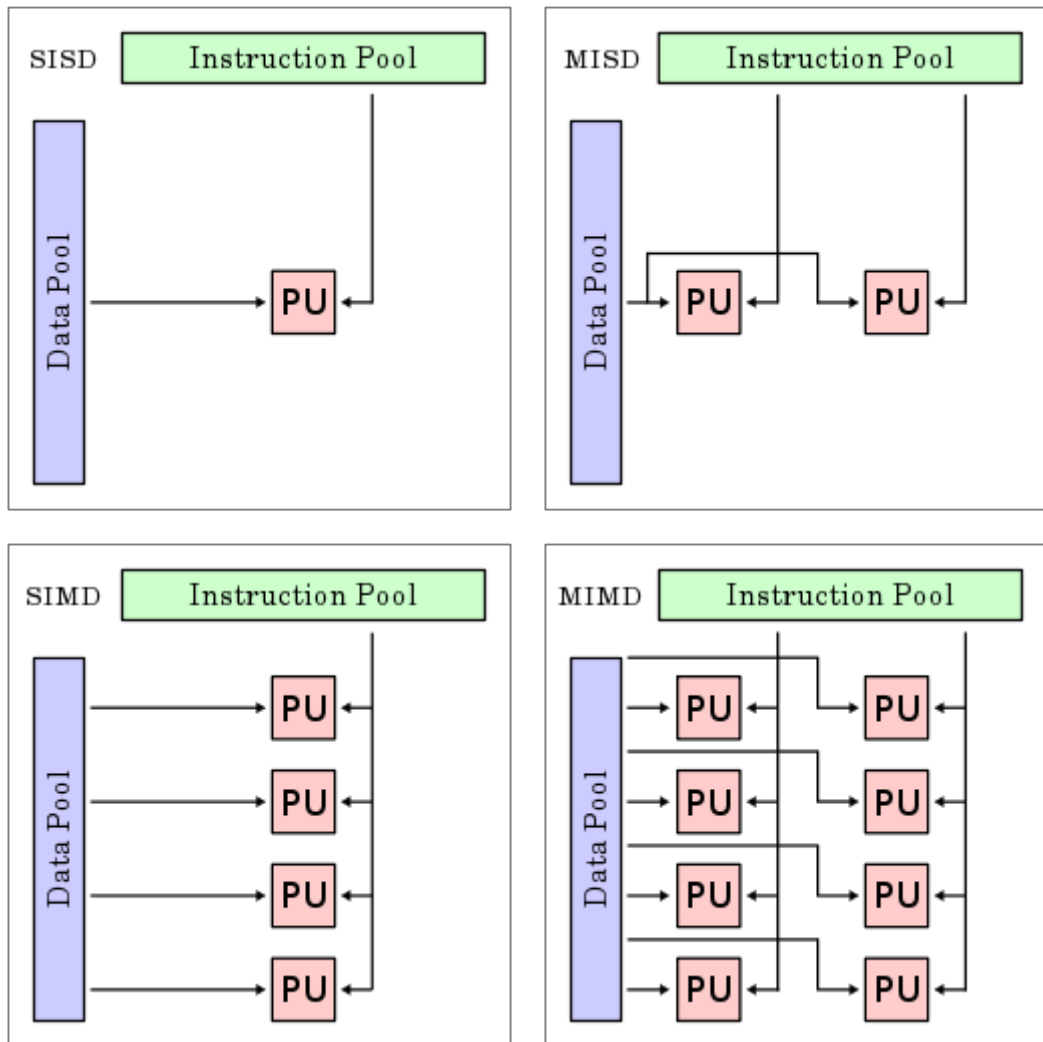
Классификация архитектур ВС

- **Классификация Флинна (M. J. Flynn, 1966)**

Flynn's taxonomy		
	Single instruction	Multiple instruction
Single data	SISD	MISD
Multiple data	SIMD	MIMD

- **По организации памяти:** с общей памятью (SMP/NUMA), с распределенной памятью
- **По типу поддерживаемых операндов:** скалярные и векторные системы
- **По доступности компонентов,** из которых строится система:
проприетарные (заказные), на основе общедоступного аппаратного обеспечения (commodity computing, проект Beowulf)
- ...

Классификация Флинна



- **Современные ВС нельзя однозначно классифицировать по Флинну** – современные системы мультиархитектурны (SIMD + MIMD)
- **Процессор** – скалярные + векторные инструкции (**SIMD**)
- **Вычислительный узел** – SMP/NUMA-система на базе многоядерных процессоров (**MIMD**) + графические процессоры (**SIMD**)
- **Совокупность нескольких вычислительных узлов** – **MIMD**

Рейтинги мощнейших ВС

1. www.top500.org – решение системы линейных алгебраических уравнений методом LU-факторизации (High-Performance Linpack, FLOPS – Floating-point Operations Per Seconds)
2. www.graph500.org – алгоритмы на графах (построение графа, обход в ширину, TEPS – Traversed Edges Per Second)
3. www.green500.org – главный критерий – энергоэффективность (объем потребляемой электроэнергии, kW)
4. <http://top50.supercomputers.ru> – рейтинг мощнейших вычислительных систем СНГ (тест High-Performance Linpack)
5. **Как создать свой тест производительности?**

	NAME	SPECS	SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Opteron 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	17.2	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	10.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	8.59	3.95

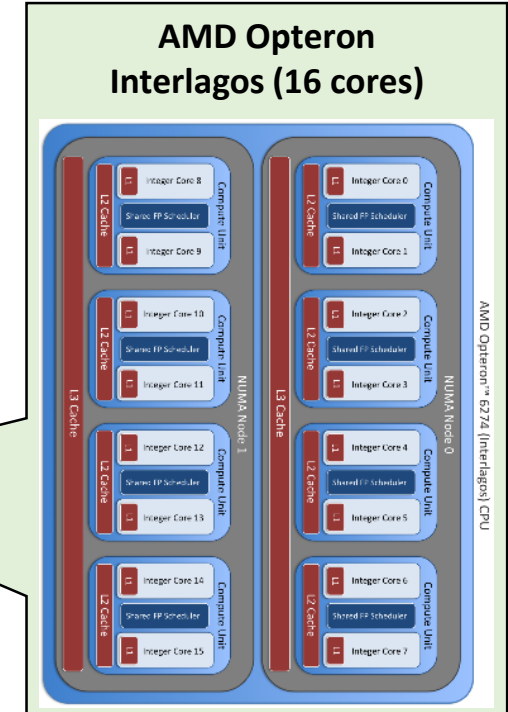
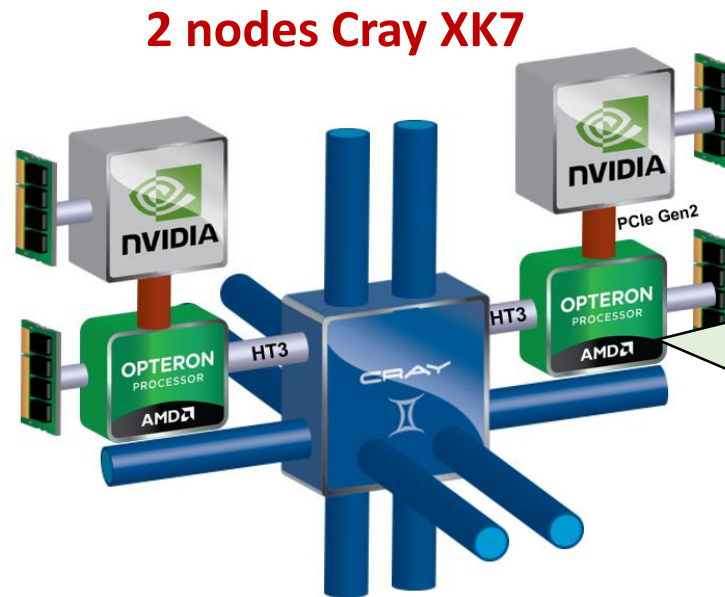
- Среднее количество вычислительных ядер в системе: **46 288**
- Среднее количество ядер на сокет (процессор): **9** (4, 6, 8, 10, 12, 14, 16)
- Среднее энергопотребление: **836** kW
- Коммуникационная сеть: Infiniband (44%), Gigabit Ethernet (25%), 10 Gigabit Ethernet (15%), Custom (10%), Cray (3%)
- Процессоры: Intel (> 80%), IBM Power, AMD Opteron, SPARC64, ShenWei, NEC
- Ускорители (14% систем): Intel Xeon Phi, NVIDIA GPU, ATI/AMD GPU, PESY-SC
- Операционная система: GNU/Linux (96%), IBM AIX (11 шт.), Microsoft (1 шт.)

	NAME	SPECS	SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Opteron 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	15.3	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	12.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	796,544	11.59	3.95

- Среднее количество вычислительных ядер в системе: 400 000
- Среднее количество ядер на сокет (процессор): 9 (4, 6, 8, 12, 16, 20, 24, 32, 40, 48, 60, 72, 80, 96, 120, 144, 160, 192, 240, 288, 320, 360, 400, 480, 576, 640, 720, 800, 864, 960, 1024, 1152, 1280, 1440, 1600, 1728, 1920, 2048, 2304, 2560, 2880, 3072, 3200, 3600, 4096, 4608, 5120, 5760, 6144, 6400, 7168, 7680, 8192, 8640, 9216, 9600, 10240, 10800, 11520, 12288, 13000, 13824, 14400, 15360, 16384, 17280, 18432, 19200, 20480, 21504, 22720, 24000, 25000, 26112, 27456, 28800, 30000, 31360, 32768, 34200, 35776, 37200, 38800, 40320, 41920, 43500, 45120, 46800, 48480, 50176, 51900, 53600, 55376, 57120, 58960, 60800, 62720, 64640, 66640, 68640, 70640, 72640, 74640, 76640, 78640, 80640, 82640, 84640, 86640, 88640, 90640, 92640, 94640, 96640, 98640, 100640, 102640, 104640, 106640, 108640, 110640, 112640, 114640, 116640, 118640, 120640, 122640, 124640, 126640, 128640, 130640, 132640, 134640, 136640, 138640, 140640, 142640, 144640, 146640, 148640, 150640, 152640, 154640, 156640, 158640, 160640, 162640, 164640, 166640, 168640, 170640, 172640, 174640, 176640, 178640, 180640, 182640, 184640, 186640, 188640, 190640, 192640, 194640, 196640, 198640, 200640, 202640, 204640, 206640, 208640, 210640, 212640, 214640, 216640, 218640, 220640, 222640, 224640, 226640, 228640, 230640, 232640, 234640, 236640, 238640, 240640, 242640, 244640, 246640, 248640, 250640, 252640, 254640, 256640, 258640, 260640, 262640, 264640, 266640, 268640, 270640, 272640, 274640, 276640, 278640, 280640, 282640, 284640, 286640, 288640, 290640, 292640, 294640, 296640, 298640, 300640, 302640, 304640, 306640, 308640, 310640, 312640, 314640, 316640, 318640, 320640, 322640, 324640, 326640, 328640, 330640, 332640, 334640, 336640, 338640, 340640, 342640, 344640, 346640, 348640, 350640, 352640, 354640, 356640, 358640, 360640, 362640, 364640, 366640, 368640, 370640, 372640, 374640, 376640, 378640, 380640, 382640, 384640, 386640, 388640, 390640, 392640, 394640, 396640, 398640, 400640, 402640, 404640, 406640, 408640, 410640, 412640, 414640, 416640, 418640, 420640, 422640, 424640, 426640, 428640, 430640, 432640, 434640, 436640, 438640, 440640, 442640, 444640, 446640, 448640, 450640, 452640, 454640, 456640, 458640, 460640, 462640, 464640, 466640, 468640, 470640, 472640, 474640, 476640, 478640, 480640, 482640, 484640, 486640, 488640, 490640, 492640, 494640, 496640, 498640, 500640, 502640, 504640, 506640, 508640, 510640, 512640, 514640, 516640, 518640, 520640, 522640, 524640, 526640, 528640, 530640, 532640, 534640, 536640, 538640, 540640, 542640, 544640, 546640, 548640, 550640, 552640, 554640, 556640, 558640, 560640, 562640, 564640, 566640, 568640, 570640, 572640, 574640, 576640, 578640, 580640, 582640, 584640, 586640, 588640, 590640, 592640, 594640, 596640, 598640, 600640, 602640, 604640, 606640, 608640, 610640, 612640, 614640, 616640, 618640, 620640, 622640, 624640, 626640, 628640, 630640, 632640, 634640, 636640, 638640, 640640, 642640, 644640, 646640, 648640, 650640, 652640, 654640, 656640, 658640, 660640, 662640, 664640, 666640, 668640, 670640, 672640, 674640, 676640, 678640, 680640, 682640, 684640, 686640, 688640, 690640, 692640, 694640, 696640, 698640, 700640, 702640, 704640, 706640, 708640, 710640, 712640, 714640, 716640, 718640, 720640, 722640, 724640, 726640, 728640, 730640, 732640, 734640, 736640, 738640, 740640, 742640, 744640, 746640, 748640, 750640, 752640, 754640, 756640, 758640, 760640, 762640, 764640, 766640, 768640, 770640, 772640, 774640, 776640, 778640, 780640, 782640, 784640, 786640, 788640, 790640, 792640, 794640, 796640, 798640, 800640, 802640, 804640, 806640, 808640, 810640, 812640, 814640, 816640, 818640, 820640, 822640, 824640, 826640, 828640, 830640, 832640, 834640, 836640, 838640, 840640, 842640, 844640, 846640, 848640, 850640, 852640, 854640, 856640, 858640, 860640, 862640, 864640, 866640, 868640, 870640, 872640, 874640, 876640, 878640, 880640, 882640, 884640, 886640, 888640, 890640, 892640, 894640, 896640, 898640, 900640, 902640, 904640, 906640, 908640, 910640, 912640, 914640, 916640, 918640, 920640, 922640, 924640, 926640, 928640, 930640, 932640, 934640, 936640, 938640, 940640, 942640, 944640, 946640, 948640, 950640, 952640, 954640, 956640, 958640, 960640, 962640, 964640, 966640, 968640, 970640, 972640, 974640, 976640, 978640, 980640, 982640, 984640, 986640, 988640, 990640, 992640, 994640, 996640, 998640, 1000640, 1002640, 1004640, 1006640, 1008640, 1010640, 1012640, 1014640, 1016640, 1018640, 1020640, 1022640, 1024640, 1026640, 1028640, 1030640, 1032640, 1034640, 1036640, 1038640, 1040640, 1042640, 1044640, 1046640, 1048640, 1050640, 1052640, 1054640, 1056640, 1058640, 1060640, 1062640, 1064640, 1066640, 1068640, 1070640, 1072640, 1074640, 1076640, 1078640, 1080640, 1082640, 1084640, 1086640, 1088640, 1090640, 1092640, 1094640, 1096640, 1098640, 1100640, 1102640, 1104640, 1106640, 1108640, 1110640, 1112640, 1114640, 1116640, 1118640, 1120640, 1122640, 1124640, 1126640, 1128640, 1130640, 1132640, 1134640, 1136640, 1138640, 1140640, 1142640, 1144640, 1146640, 1148640, 1150640, 1152640, 1154640, 1156640, 1158640, 1160640, 1162640, 1164640, 1166640, 1168640, 1170640, 1172640, 1174640, 1176640, 1178640, 1180640, 1182640, 1184640, 1186640, 1188640, 1190640, 1192640, 1194640, 1196640, 1198640, 1200640, 1202640, 1204640, 1206640, 1208640, 1210640, 1212640, 1214640, 1216640, 1218640, 1220640, 1222640, 1224640, 1226640, 1228640, 1230640, 1232640, 1234640, 1236640, 1238640, 1240640, 1242640, 1244640, 1246640, 1248640, 1250640, 1252640, 1254640, 1256640, 1258640, 1260640, 1262640, 1264640, 1266640, 1268640, 1270640, 1272640, 1274640, 1276640, 1278640, 1280640, 1282640, 1284640, 1286640, 1288640, 1290640, 1292640, 1294640, 1296640, 1298640, 1300640, 1302640, 1304640, 1306640, 1308640, 1310640, 1312640, 1314640, 1316640, 1318640, 1320640, 1322640, 1324640, 1326640, 1328640, 1330640, 1332640, 1334640, 1336640, 1338640, 1340640, 1342640, 1344640, 1346640, 1348640, 1350640, 1352640, 1354640, 1356640, 1358640, 1360640, 1362640, 1364640, 1366640, 1368640, 1370640, 1372640, 1374640, 1376640, 1378640, 1380640, 1382640, 1384640, 1386640, 1388640, 1390640, 1392640, 1394640, 1396640, 1398640, 1400640, 1402640, 1404640, 1406640, 1408640, 1410640, 1412640, 1414640, 1416640, 1418640, 1420640, 1422640, 1424640, 1426640, 1428640, 1430640, 1432640, 1434640, 1436640, 1438640, 1440640, 1442640, 1444640, 1446640, 1448640, 1450640, 1452640, 1454640, 1456640, 1458640, 1460640, 1462640, 1464640, 1466640, 1468640, 1470640, 1472640, 1474640, 1476640, 1478640, 1480640, 1482640, 1484640, 1486640, 1488640, 1490640, 1492640, 1494640, 1496640, 1498640, 1500640, 1502640, 1504640, 1506640, 1508640, 1510640, 1512640, 1514640, 1516640, 1518640, 1520640, 1522640, 1524640, 1526640, 1528640, 1530640, 1532640, 1534640, 1536640, 1538640, 1540640, 1542640, 1544640, 1546640, 1548640, 1550640, 1552640, 1554640, 1556640, 1558640, 1560640, 1562640, 1564640, 1566640, 1568640, 1570640, 1572640, 1574640, 1576640, 1578640, 1580640, 1582640, 1584640, 1586640, 1588640, 1590640, 1592640, 1594640, 1596640, 1598640, 1600640, 1602640, 1604640, 1606640, 1608640, 1610640, 1612640, 1614640, 1616640, 1618640, 1620640, 1622640, 1624640, 1626640, 1628640, 1630640, 1632640, 1634640, 1636640, 1638640, 1640640, 1642640, 1644640, 1646640, 1648640, 1650640, 1652640, 1654640, 1656640, 1658640, 1660640, 1662640, 1664640, 1666640, 1668640, 1670640, 1672640, 1674640, 1676640, 1678640, 1680640, 1682640, 1684640, 1686640, 1688640, 1690640, 1692640, 1694640, 1696640, 1698640, 1700640, 1702640, 1704640, 1706640, 1708640, 1710640, 1712640, 1714640, 1716640, 1718640, 1720640, 1722640, 1724640, 1726640, 1728640, 1730640, 1732640, 1734640, 1736640, 1738640, 1740640, 1742640, 1744640, 1746640, 1748640, 1750640, 1752640, 1754640, 1756640, 1758640, 1760640, 1762640, 1764640, 1766640, 1768640, 1770640, 1772640, 1774640, 1776640, 1778640, 1780640, 1782640, 1784640, 1786640, 1788640, 1790640, 1792640, 1794640, 1796640, 1798640, 1800640, 1802640, 1804640, 1806640, 1808640, 1810640, 1812640, 1814640, 1816640, 1818640, 1820640, 1822640, 1824640, 1826640, 1828640, 1830640, 1832640, 1834640, 1836640, 1838640, 1840640, 1842640, 1844640, 1846640, 1848640, 1850640, 1852640, 1854640, 1856640, 1858640, 1860640, 1862640, 1864640, 1866640, 1868640, 1870640, 1872640, 1874640, 1876640, 1878640, 1880640, 1882640, 1884640, 1886640, 1888640, 1890640, 1892640, 1894640, 1896640, 1898640, 1900640, 1902640, 1904640, 1906640, 1908640, 1910640, 1912640, 1914640, 1916640, 1918640, 1920640, 1922640, 1924640, 1926640, 1928640, 1930640, 1932640, 1934640, 1936640, 1938640, 1940640, 1942640, 1944640, 1946640, 1948640, 1950640, 1952640, 1954640, 1956640, 1958640, 1960640, 1962640, 1964640, 1966640, 1968640, 1970640, 1972640, 1974640, 1976640, 1978640, 1980640, 1982640, 1984640, 1986640, 1988640, 1990640, 1992640, 1994640, 1996640, 1998640, 2000640, 2002640, 2004640, 2006640, 2008640, 2010640, 2012640, 2014640, 2016640, 2018640, 2020640, 2022640, 2024640, 2026640, 2028640, 2030640, 2032640, 2034640, 2036640, 2038640, 2040640, 2042640, 2044640, 2046640, 2048640, 2050640, 2052640, 2054640, 2056640, 2058640, 2060640, 2062640, 2064640, 2066640, 2068640, 2070640, 2072640, 2074640, 2076640, 2078640, 2080640, 2082640, 2084640, 2086640, 2088640, 2090640, 2092640, 2094640, 2096640, 2098640, 2100640, 2102640, 2104640, 2106640, 2108640, 2110640, 2112640, 2114640, 2116640, 2118640, 2120640, 2122640, 2124640, 2126640, 2128640, 2130640, 2132640, 2134640, 2136640, 2138640, 2140640, 2142640, 2144640, 2146640, 2148640, 2150640, 2152640, 2154640, 2156640, 2158640, 2160640, 2162640, 2164640, 2166640, 2168640, 2170640, 2172640, 2174640, 2176640, 2178640, 2180640, 2182640, 2184640, 2186640, 2188640, 2190640, 2192640, 2194640, 2196640, 2198640, 2200640, 2202640, 2204640, 2206640, 2208640, 2210640, 2212640, 2214640, 2216640, 2218640, 2220640, 2222640, 2224640, 2226640, 2228640, 2230640, 2232640, 2234640, 2236640, 2238640, 2240640, 2242640, 2244640, 2246640, 2248640, 2250640, 2252640, 2254640, 2256640, 2258640, 2260640, 2262640, 2264640, 2266640, 2268640, 2270640, 2272640, 2274640, 2276640, 2278640, 2280640, 2282640, 2284640, 2286640, 2288640, 2290640, 2292640, 2294640, 2296640, 2298640, 2300640, 2302640, 2304640, 2306640, 2308640, 2310640, 2312640, 2314640, 2316640, 2318640, 2320640, 2322640, 2324640, 2326640, 2328640, 2330640, 2332640, 2334640, 2336640, 2338640, 2340640, 2342640, 2344640, 2346640, 2348640, 2350640, 2352640, 2354640, 2356640, 2358640, 2360640, 2362640, 2364640, 2366640, 2368640, 2370640, 2372640, 2374640, 2376640, 2378640, 2380640, 2382640, 2384640, 2386640, 2388640, 2390640, 2392640, 2394640, 2396640, 2398640, 2400640, 2402640, 2404640, 2406640, 2408640, 2410640, 2412640, 2414640, 2416640, 2418640, 2420640, 2422640, 2424640, 2426640, 2428640, 2430640, 2432640, 2434640, 2436640, 2438640, 2440640, 2442640, 2444640, 2446640, 2448640, 2450640, 2452640, 2454640, 2456640, 2458640, 2460640, 2462640, 2464640, 2466640, 2468640, 2470640, 2472640, 2474640, 2476640, 2478640, 2480640, 2482640, 2484640, 2486640, 2488640, 2490640, 2492640, 2494640, 2496640, 2498640, 2500640, 2502640, 2504640, 2506640, 2508640, 2510640, 2512640, 2514640, 2516640, 2518640, 2520640, 2522640, 2524640, 2526640, 2528640, 2530640, 2532640, 2534640, 2536640, 2538640, 2540640, 2542640, 2544640, 2546640, 2548640, 2550640, 2552640, 2554640, 2556640, 2558640, 2560640, 2562640, 2564640, 2566640, 2568640, 2570640, 2572640, 2574640, 2576640, 2578640, 2580640, 2582640, 2584640, 2586640, 2588640, 2590640, 2592640, 2594640, 2596640, 2598640, 2600640, 2602640, 2604640, 2606640, 2608640, 2610640, 2612640, 2614640, 2616640, 2618640, 2620640, 2622640, 2624640, 2626640, 2628640, 2630640, 2632640, 2634640, 2636640, 263864

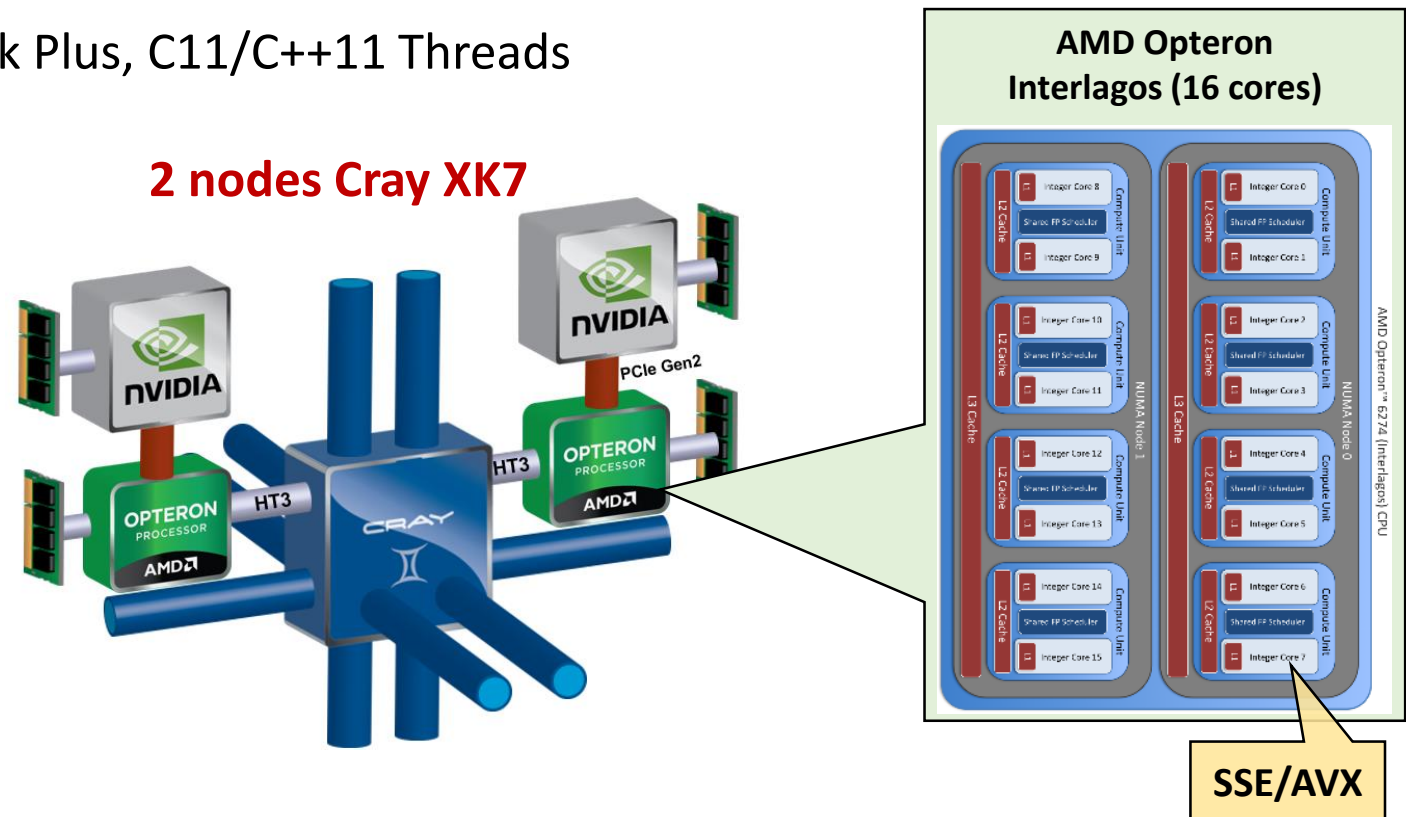
Мультиархитектура современных ВС

- Разработка эффективных параллельных программ для мультиархитектурных ВС требует владения стеком технологий
 - ❑ **Internode communications:**
[MPI](#), Cray Chapel, IBM X10, Shmem, Unified Parallel C, Coarray Fortran, Global Arrays
 - ❑ **Multithreading:** [OpenMP](#), Intel TBB/Cilk Plus, C11/C++11 Threads
 - ❑ **GPU:** [NVIDIA CUDA](#), OpenCL, OpenACC, OpenMP 4.0
 - ❑ **Vectorization (SIMD):**
[SSE/AVX](#), AltiVec



Мультиархитектура современных ВС

- Разработка эффективных параллельных программ для мультиархитектурных ВС требует владения стеком технологий
 - ❑ **Internode communications:**
[MPI](#), Cray Chapel, IBM X10, Shmem, Unified Parallel C, Coarray Fortran, Global Arrays
 - ❑ **Multithreading:** [OpenMP](#), Intel TBB/Cilk Plus, C11/C++11 Threads
 - ❑ **GPU:** [NVIDIA CUDA](#), OpenCL, OpenACC, OpenMP 4.0
 - ❑ **Vectorization (SIMD):**
[SSE/AVX](#), AltiVec



Мультиархитектура современных ВС

- Разработка эффективных параллельных программ для мультиархитектурных ВС требует владения стеком технологий

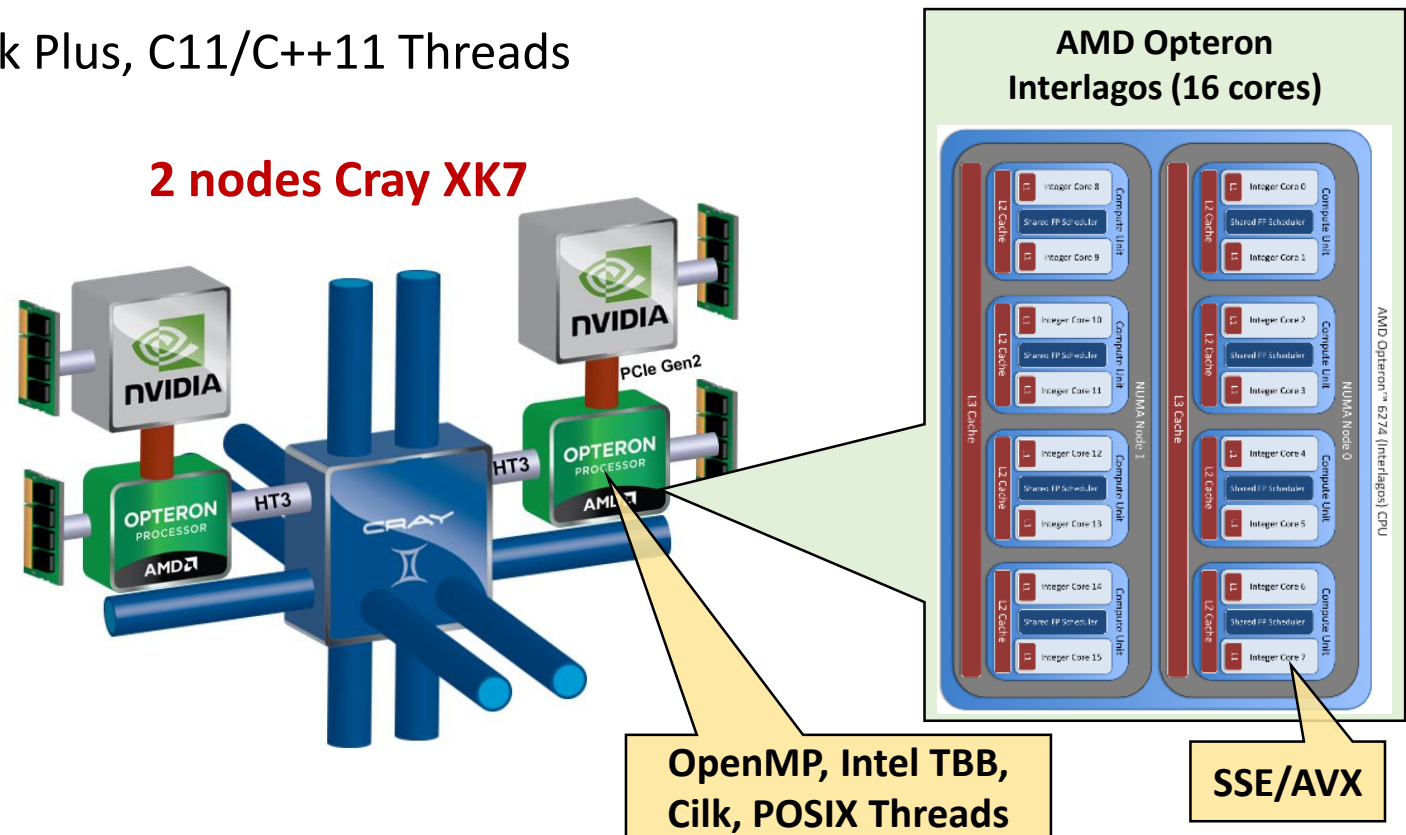
- ❑ Internode communications:

[MPI](#), Cray Chapel, IBM X10, Shmem, Unified Parallel C, Coarray Fortran, Global Arrays

- ❑ Multithreading: [OpenMP](#), Intel TBB/Cilk Plus, C11/C++11 Threads

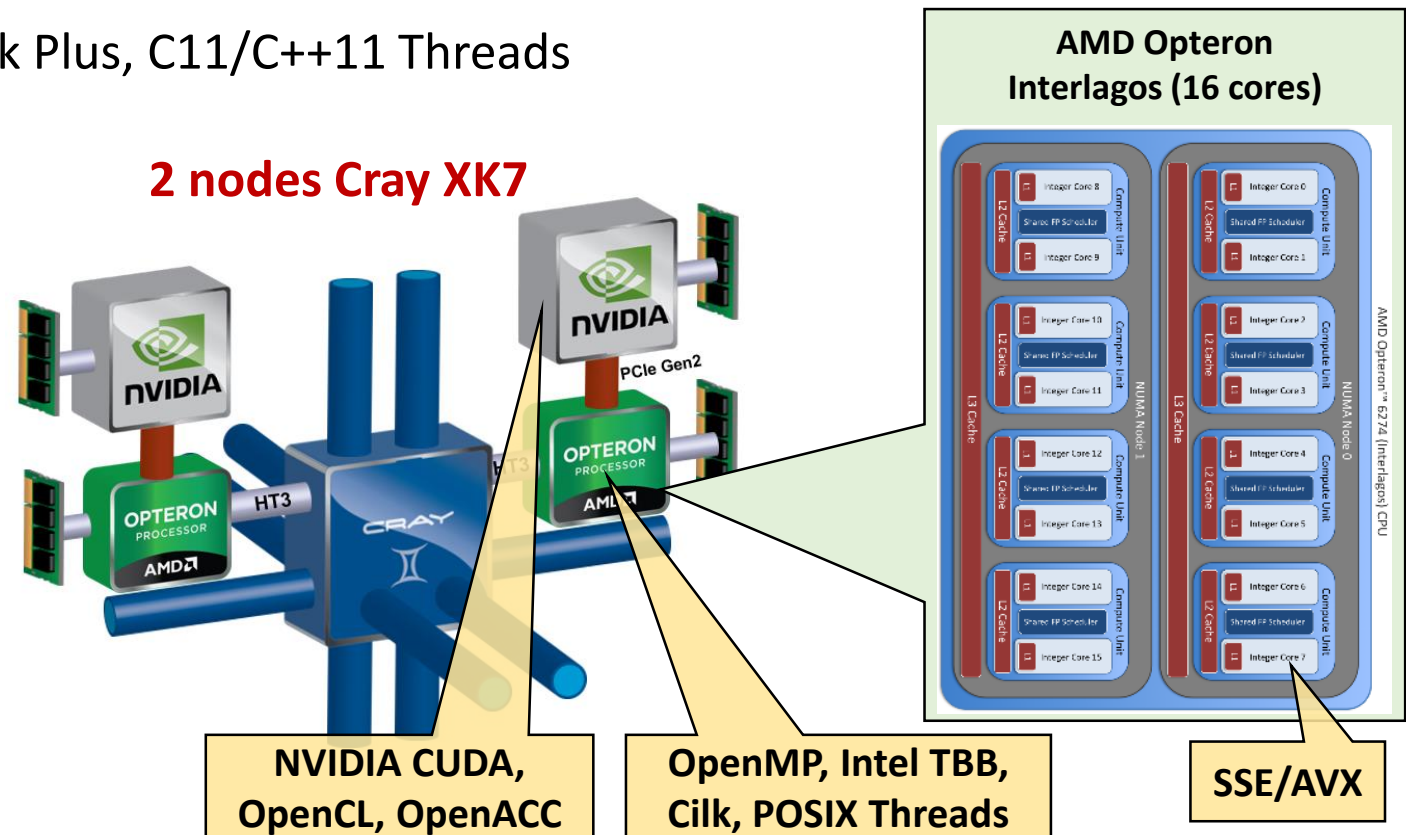
- ❑ GPU: [NVIDIA CUDA](#), OpenCL, OpenACC, OpenMP 4.0

- ❑ Vectorization (SIMD): [SSE/AVX](#), AltiVec



Мультиархитектура современных ВС

- Разработка эффективных параллельных программ для мультиархитектурных ВС требует владения стеком технологий
 - ❑ **Internode communications:**
[MPI](#), Cray Chapel, IBM X10, Shmem, Unified Parallel C, Coarray Fortran, Global Arrays
 - ❑ **Multithreading:** [OpenMP](#), Intel TBB/Cilk Plus, C11/C++11 Threads
 - ❑ **GPU:** [NVIDIA CUDA](#), OpenCL, OpenACC, OpenMP 4.0
 - ❑ **Vectorization (SIMD):**
[SSE/AVX](#), AltiVec



Мультиархитектура современных ВС

- Разработка эффективных параллельных программ для мультиархитектурных ВС требует владения стеком технологий

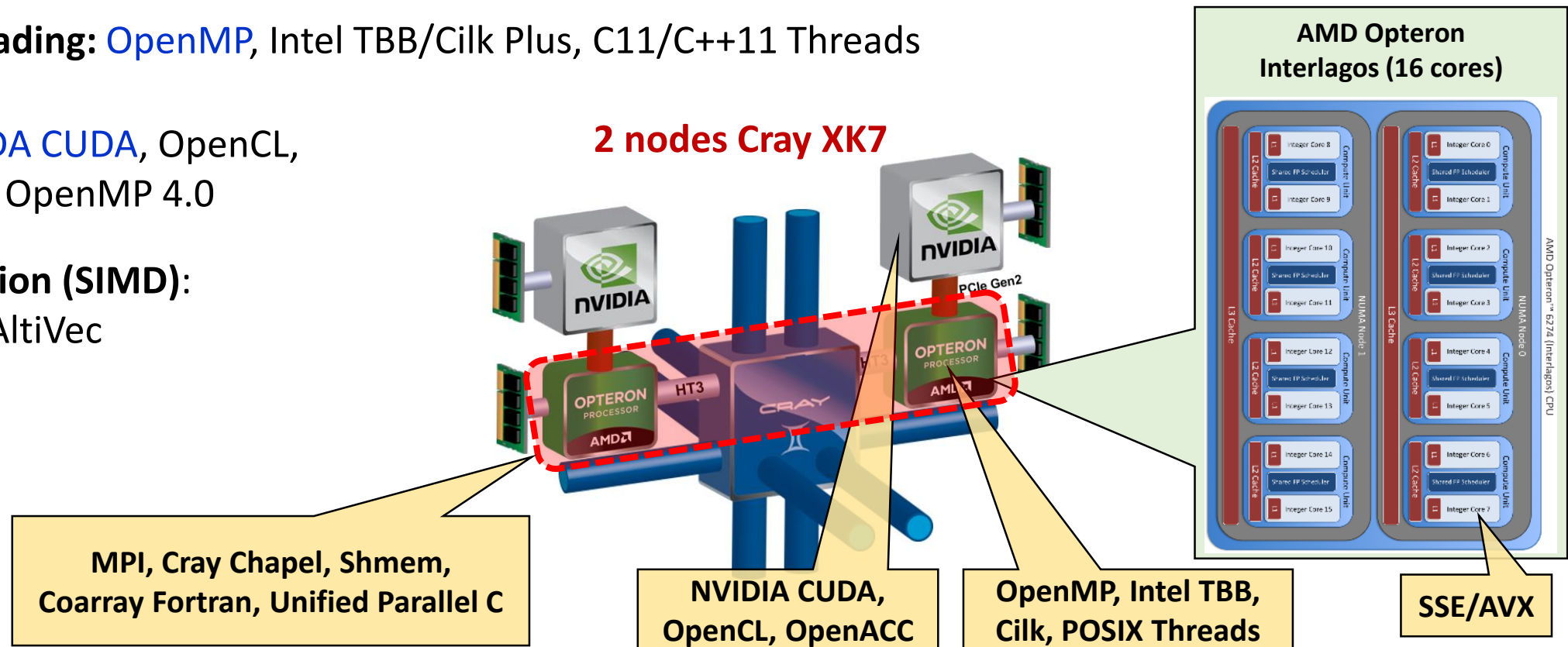
- ❑ Internode communications:

[MPI](#), Cray Chapel, IBM X10, Shmem, Unified Parallel C, Coarray Fortran, Global Arrays

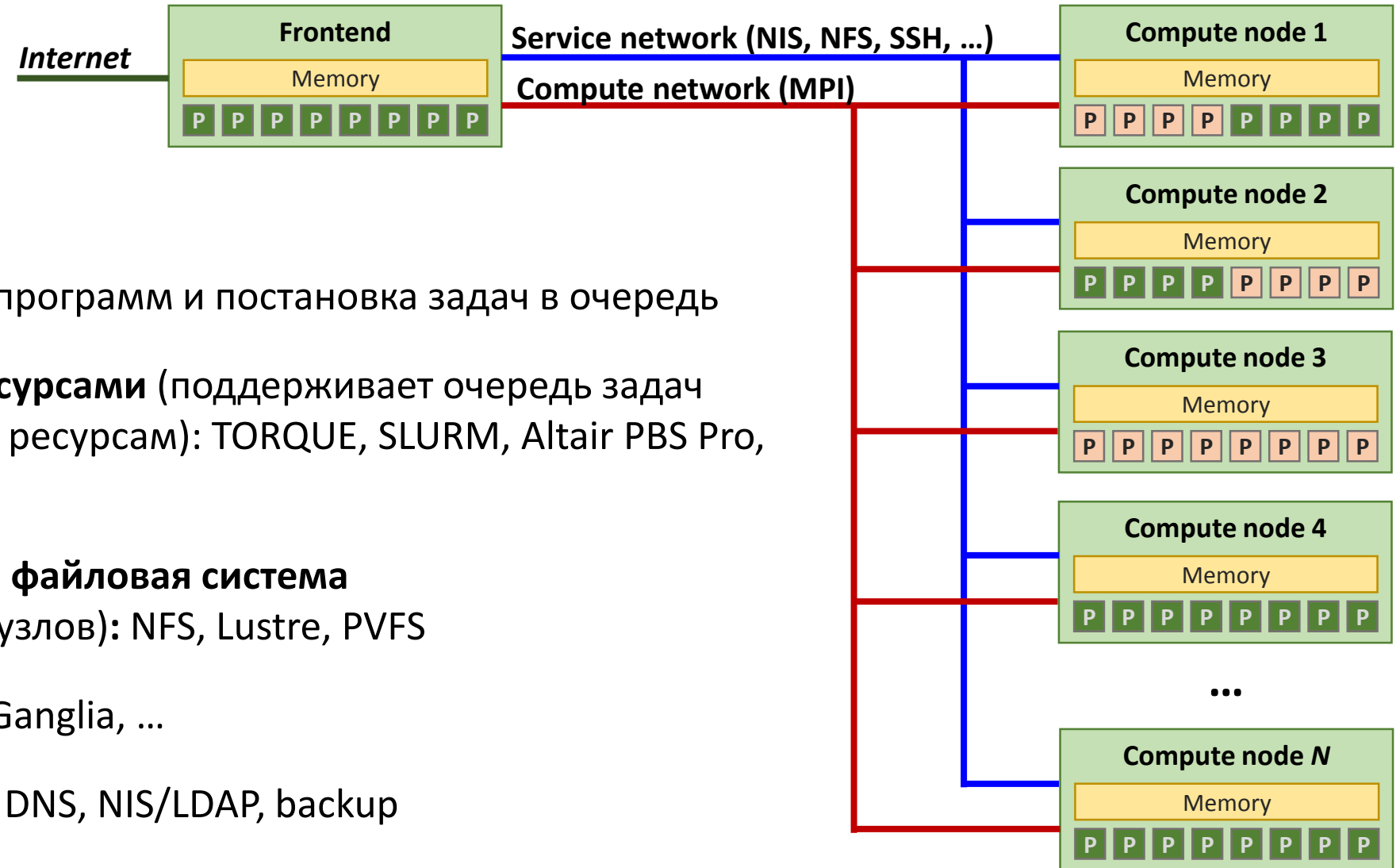
- ❑ Multithreading: [OpenMP](#), Intel TBB/Cilk Plus, C11/C++11 Threads

- ❑ GPU: [NVIDIA CUDA](#), OpenCL, OpenACC, OpenMP 4.0

- ❑ Vectorization (SIMD): [SSE/AVX](#), AltiVec



Программное обеспечение вычислительных кластеров



- **Frontend** – компиляция программ и постановка задач в очередь
- **Система управления ресурсами** (поддерживает очередь задач и контролирует доступ к ресурсам): TORQUE, SLURM, Altair PBS Pro, IBM Load Leveler, ...
- **Сетевая (параллельная) файловая система** (доступ к /home со всех узлов): NFS, Lustre, PVFS
- **Система мониторинга**: Ganglia, ...
- Сетевые сервисы: DHCP, DNS, NIS/LDAP, backup

Многопроцессорные ВС с общей памятью (SMP, NUMA)

- **Плюсы**

- ☐ Привычная модель программирования (потоки, процессы)
- ☐ Высокая скорость обмена данными между потоками/процессами

- **Минусы**

- ☐ Синхронизация при доступе к общим данным (критические секции)
- ☐ Когерентность кэшей, ложное разделение данных
- ☐ Относительно низкая масштабируемость
(как правило, 4 – 16 процессоров на узле)
- ☐ Трудоемкая организация эффективного использования памяти в NUMA-системах

Многопроцессорные ВС с распределенной памятью

- **Плюсы**

- ☐ Высокая масштабируемость (сотни, тысячи и миллионы процессорных ядер)
- ☐ Меньше проблем с синхронизацией (у каждого узла/сервера своя память)
- ☐ Декомпозиция на крупные подзадачи

- **Минусы**

- ☐ Необходимость использования передачи сообщений (message passing)
- ☐ Высокие временные задержки и низкая пропускная способность (все взаимодействия по сети – Gigabit Ethernet, Infiniband)
- ☐ Неоднородность, отказы узлов

Теоретические основы параллельных вычислений

Параллельные вычисления (Parallel Computing)

- **Разработка параллельного алгоритма**
 - ❑ Поиск параллелизма в известном последовательном алгоритме, его модификация или создание нового алгоритма
 - ❑ Декомпозиция задачи на подзадачи, которые могут выполняться параллельно
 - ❑ Анализ зависимостей между подзадачами
- Параллельная версия самого эффективного последовательного алгоритма решения задачи не обязательно будет самой эффективной параллельной реализацией

Параллельные вычисления (Parallel Computing)

- **Реализация параллельного алгоритма в виде параллельной программы**
 - ❑ Распределение подзадач между процессорами (task mapping, load balancing)
 - ❑ Организация взаимодействия подзадач (message passing, shared data structures)
 - ❑ Учет архитектуры целевой вычислительной системы
 - ❑ Запуск, измерение и анализ показателей эффективности параллельной программы
 - ❑ Оптимизация программы

Показатели эффективности параллельных алгоритмов

- Коэффициент ускорения (Speedup)
- Коэффициент эффективности (Efficiency)
- Коэффициент накладных расходов
- Показатель равномерности загрузки параллельных ветвей (процессов, потоков)

Коэффициент ускорения (Speedup)

- Введем обозначения:
 - $T(n)$ – время выполнения последовательной программы (sequential program)
 - $T_p(n)$ – время выполнения параллельной программы (parallel program) на p процессорах

- **Коэффициент $S_p(n)$ ускорения** параллельной программ (Speedup):

$$S_p(n) = \frac{T(n)}{T_p(n)}$$

- Коэффициент ускорения $S_p(n)$ показывает во сколько раз параллельная программа выполняется на p процессорах быстрее последовательной программы при обработке одних и тех же входных данных размера n
- Как правило

$$S_p(n) \leq p$$

Коэффициент ускорения (Speedup)

- Введем обозначения:
 - $T(n)$ – время выполнения последовательной программы (sequential program)
 - $T_p(n)$ – время выполнения параллельной программы (parallel program) на p процессорах

- Коэффициент $S_p(n)$ ускорения параллельной программ (Speedup):

$$S_p(n) = \frac{T(n)}{T_p(n)}$$

- Цель распараллеливания – достичь линейного ускорения на максимально большом числе процессоров

$$S_p(n) \approx p \quad \text{или} \quad S_p(n) = \Omega(p) \quad \text{при} \quad p \rightarrow \infty$$

Коэффициент ускорения (Speedup)

- **Какое время брать за время выполнения последовательной программы?**
 - Время лучшего известного алгоритма (в смысле вычислительной сложности)?
 - Время лучшего теоретически возможного алгоритма?
- **Что считать временем выполнения $T_p(n)$ параллельной программы?**
 - Среднее время выполнения потоков программы?
 - Время выполнения потока, завершившего работу первым?
 - Время выполнения потока, завершившего работу последним?

Коэффициент ускорения (Speedup)

- Какое время брать за время выполнения последовательной программы?
 - Время лучшего известного алгоритма или время алгоритма, который подвергается распараллеливанию
- Что считать временем выполнения $T_p(n)$ параллельной программы?
 - Время выполнения потока, завершившего работу последним

Коэффициент ускорения (Speedup)

- **Коэффициент относительного ускорения** (Relative speedup) – отношения времени выполнения параллельной программы на k процессорах к времени её выполнения на p процессорах ($k < p$)

$$S_{Relative}(k, p, n) = \frac{T_k(n)}{T_p(n)}$$

- Коэффициент эффективности (Efficiency) параллельной программы

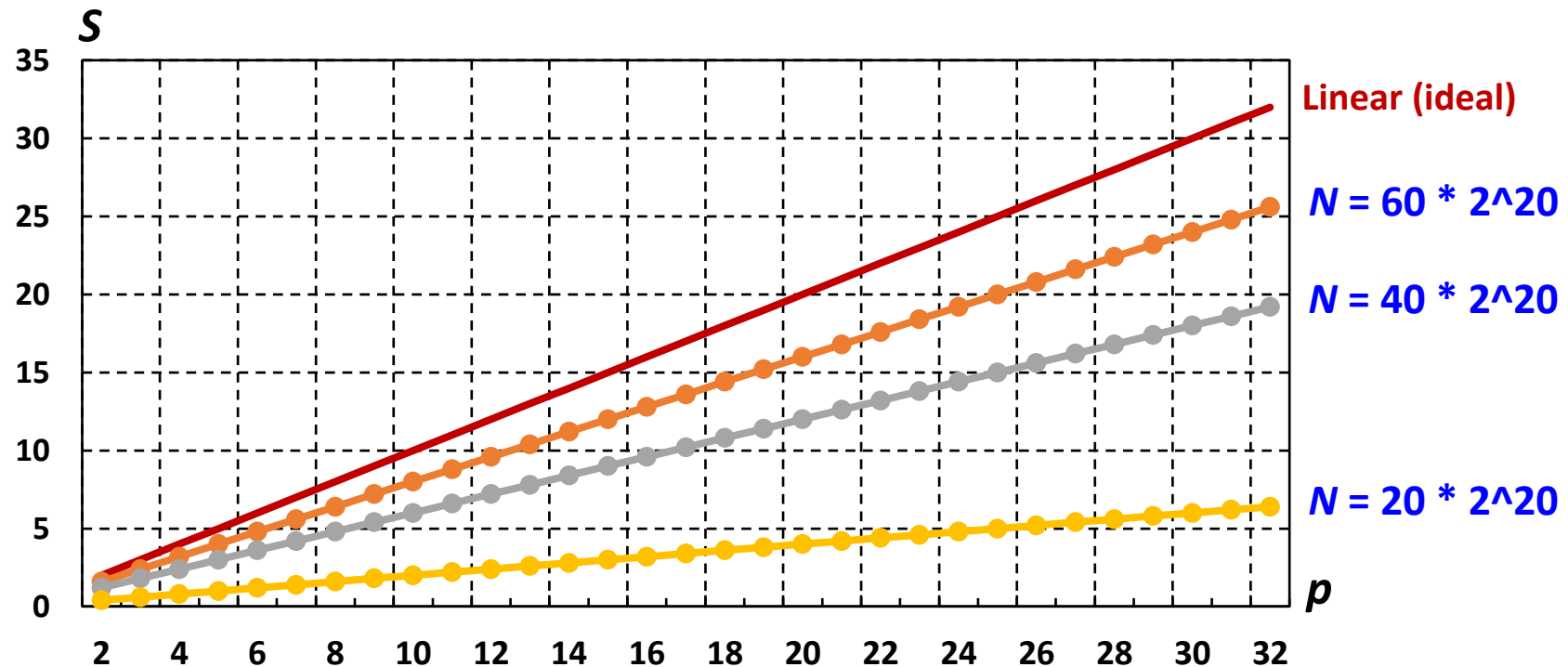
$$E_p(n) = \frac{S_p(n)}{p} = \frac{T(n)}{pT_p(n)} \in [0, 1]$$

- Коэффициент накладных расходов (Overhead)

$$\varepsilon(p, n) = \frac{T_{Sync}(p, n)}{T_{Comp}(p, n)} = \frac{T_{Total}(p, n) - T_{Comp}(p, n)}{T_{Comp}(p, n)}$$

- $T_{Sync}(p, n)$ – время создания, синхронизации и взаимодействия p потоков
- $T_{Comp}(p, n)$ – время вычислений в каждом из p потоков

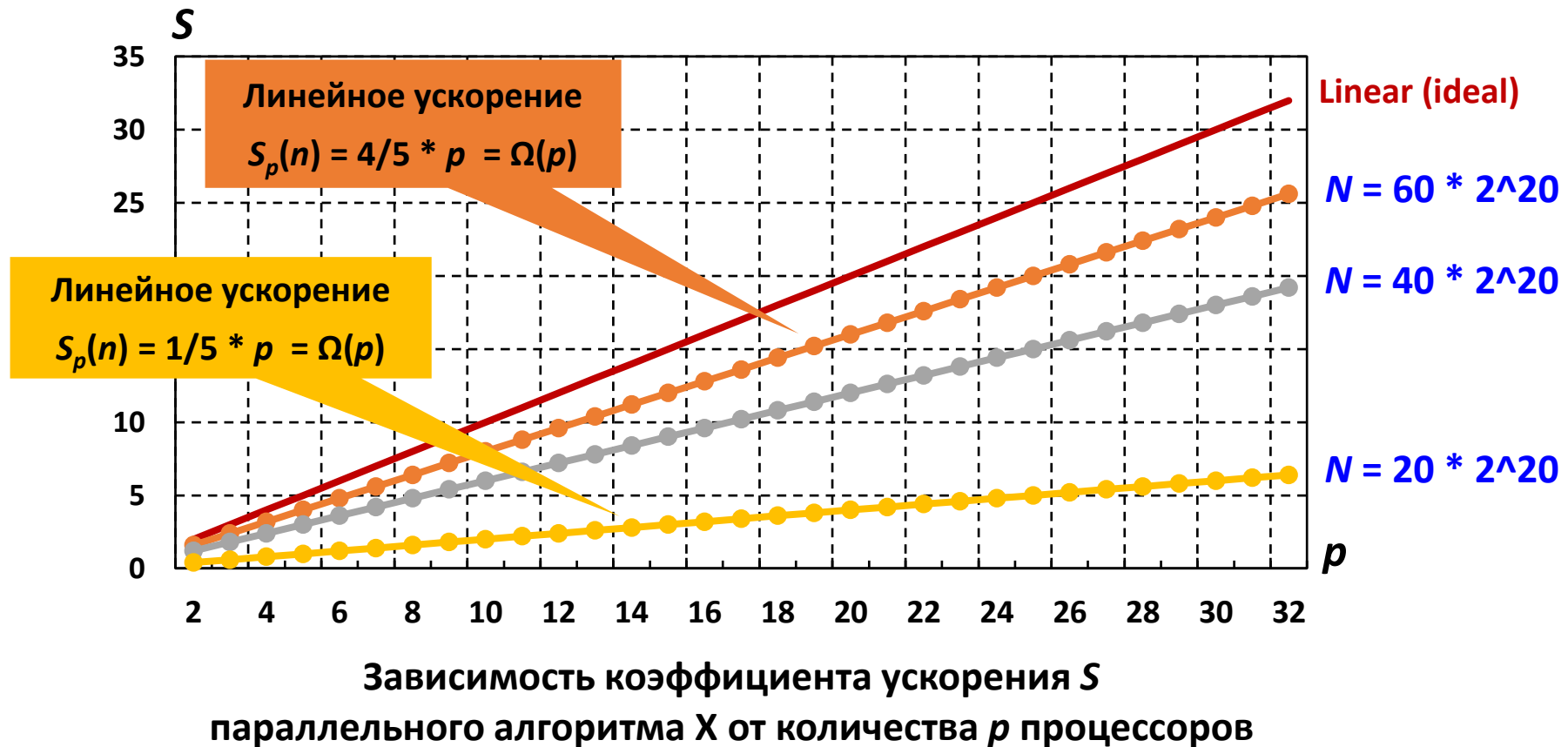
Коэффициент ускорения (Speedup)



Зависимость коэффициента ускорения S
параллельного алгоритма X от количества p процессоров

- Ускорение программы может расти с увеличением размера входных данных
- Время вычислений превосходит накладные расходы на взаимодействия потоков (управление потоками, синхронизацию, обмен сообщениями, ...)

Коэффициент ускорения (Speedup)

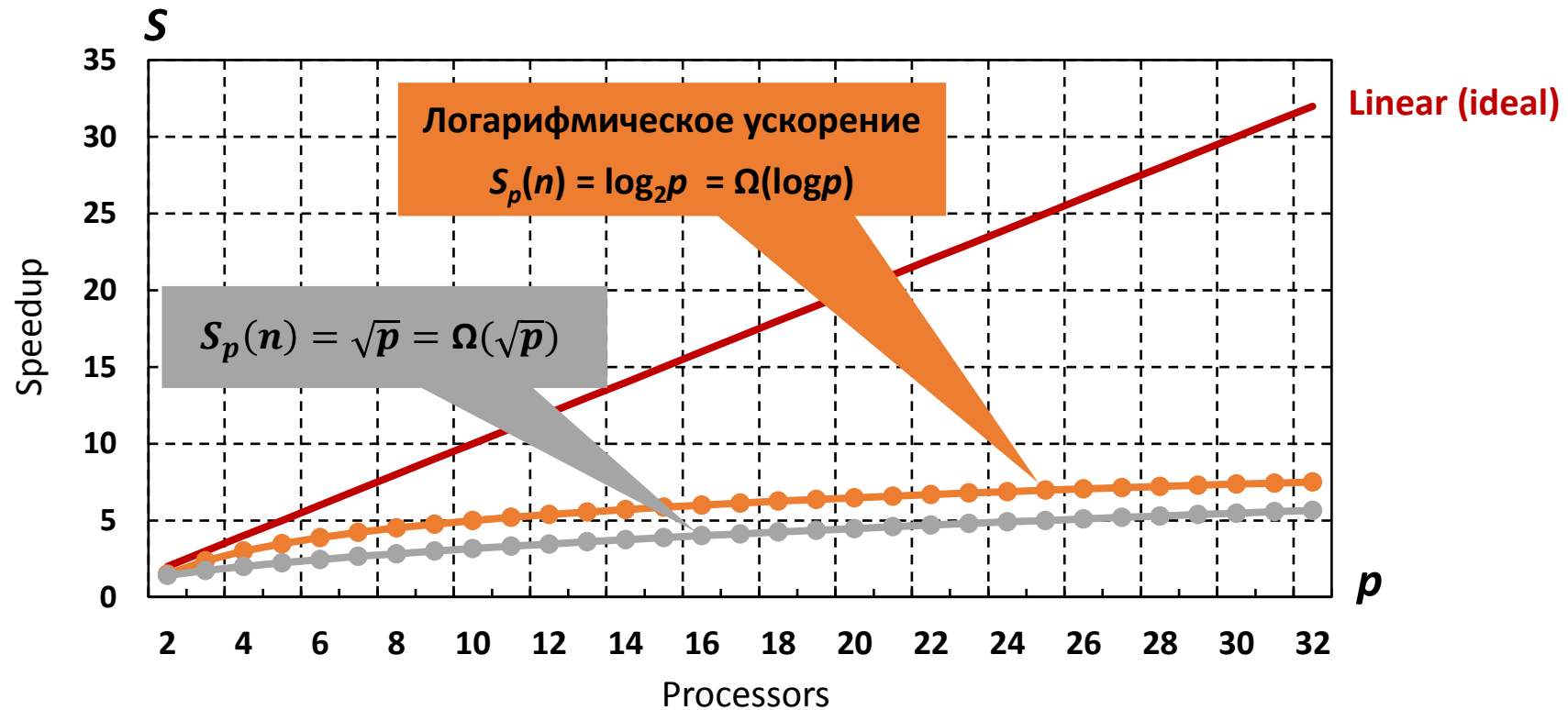


- Ускорение программы может расти с увеличением размера входных данных
- Время вычислений превосходит накладные расходы на взаимодействия потоков (управление потоками, синхронизацию, обмен сообщениями, ...)

Коэффициент ускорения (Speedup)

- Параллельная программа (алгоритм) коэффициент ускорения, которой линейной растет с увеличением p называется линейно масштабируемой или просто **масштабируемой** (scalable)
- Масштабируемая параллельная программа допускает эффективную реализацию на различном числе процессоров

Коэффициент ускорения (Speedup)



Зависимость коэффициента ускорения S параллельных алгоритмов Y и Z от количества p процессоров

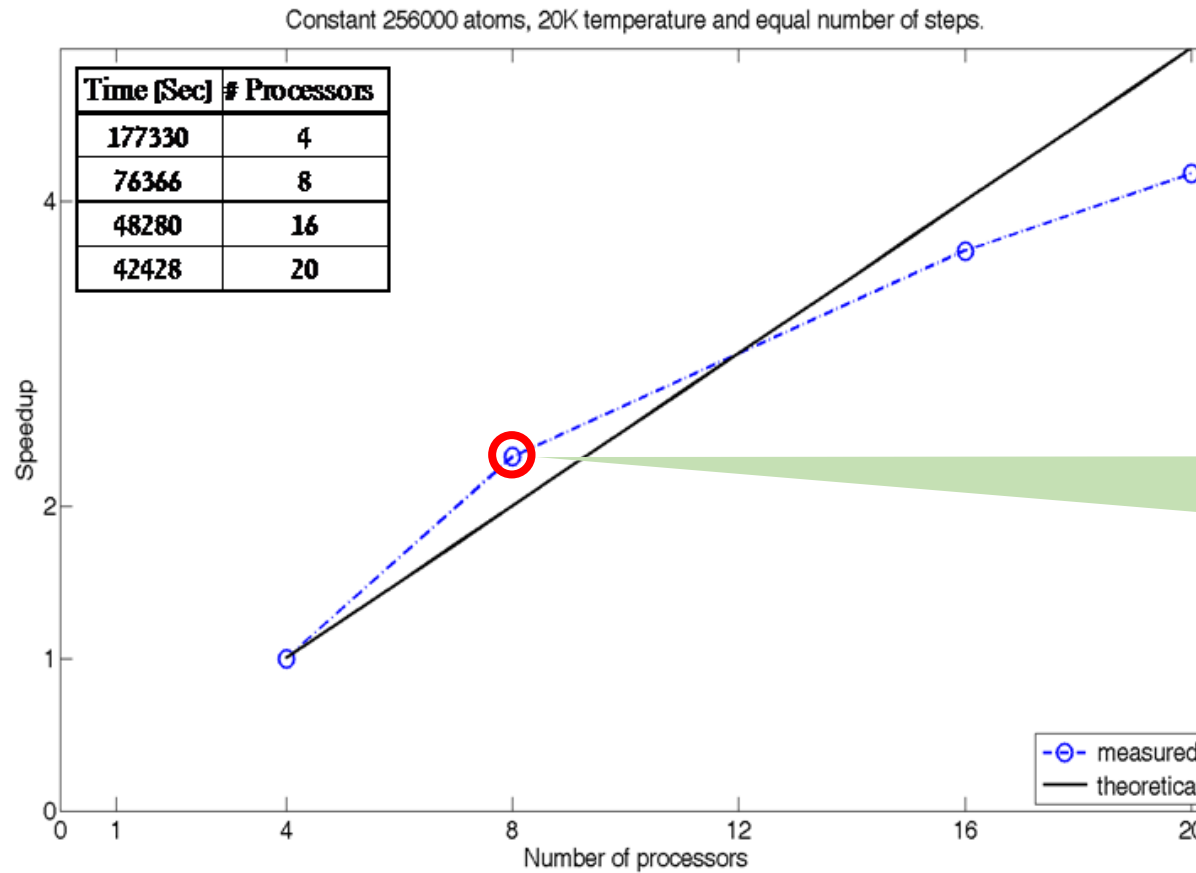
Суперлинейное ускорение (Superlinear speedup)

- Параллельная программа может характеризоваться **суперлинейным ускорением** (Superlinear speedup) – коэффициент ускорения $S_p(n)$ принимает значение больше p

$$S_p(n) > p$$

- Причина: иерархическая организация памяти:
Cache – RAM – Local disk (HDD/SSD) – Network storage
- Последовательная программа выполняется на одном процессоре и обрабатывает данные размера n
- Параллельная программа имеет p потоков на p процессорах, каждый поток работает со своей частью данных, большая часть которых может попасть в кеш-память, в результате в каждом потоке сокращается время доступа к данным
- Тот же самый эффект можно наблюдать имея два уровня иерархической памяти:
диск – память

Суперлинейное ускорение (Superlinear speedup)



Superlinear speedup

$$S_8 = \frac{T_4}{T_8} = 2.32$$

Parallel Molecular Dynamic Simulation

MPI, Spatial decomposition; Cluster nodes: 2 x AMD Opteron Dual Core; InfiniBand network

http://phycomp.technion.ac.il/~pavelba/Comp_Phys/Project/Project.html

Равномерность распределения вычислений

- По какому показателю оценивать равномерность времени выполнения потоков/процессов параллельной программы?
- Известно время выполнения потоков t_0, t_1, \dots, t_p
- Коэффициент V вариации

$$V = \frac{\sigma[t_i]}{\mu[t_i]}$$

- Отношение min/max

$$M = \frac{\min\{t_i\}}{\max\{t_i\}}$$

- Jain's fairness index

$$f = \frac{\left(\sum_{i=0}^{p-1} t_i\right)^2}{n \sum_{i=0}^{p-1} t_i^2} \in [0, 1]$$

“Последовательные” части в программах

- Инициализация и завершение работы
- Чтение входных данных и запись
- Инициализация данных
- Синхронизация, критические секции
- Пул потоков обрабатывает независимые задания
 - Извлечение заданий из очереди
 - Обработка результатов
 - Запись результатов в общую структуру данных
 - Слияние результатов из локальных структур данных

Закон Дж. Амдала (Amdahl's law)

- Пусть имеется последовательная программа с временем выполнения $T(n)$
- Обозначим:
 - $r \in [0, 1]$ – часть программы, которая может быть распараллелена (perfectly parallelized)
 - $s = 1 - r$ – часть программы, которая не может быть распараллелена (purely sequential)
- Время выполнения параллельной программы на p процессорах (время каждого потока) складывается из последовательной части s и параллельной r :

$$T_p(n) = T(n)s + \frac{T(n)}{p}r$$

- Вычислим значение коэффициент ускорения (по определению)

$$S_p(n) = \frac{T(n)}{T_p(n)} = \frac{T(n)}{T(n)s + \frac{T(n)}{p}r} = \frac{1}{s + \frac{r}{p}} = \frac{1}{(1 - r) + \frac{r}{p}}$$

- Полученная формула по значениям r и s позволяет оценить максимальное ускорение



Закон Дж. Амдала (Amdahl's law)

- Пусть имеется последовательная программа с временем выполнения $T(n)$
- Обозначим:
 - $r \in [0, 1]$ – часть программы, которая может быть распараллелена (perfectly parallelized)
 - $s = 1 - r$ – часть программы, которая не может быть распараллелена (purely sequential)
- Закон Дж. Амдала (Gene Amdahl, 1967) [1]:

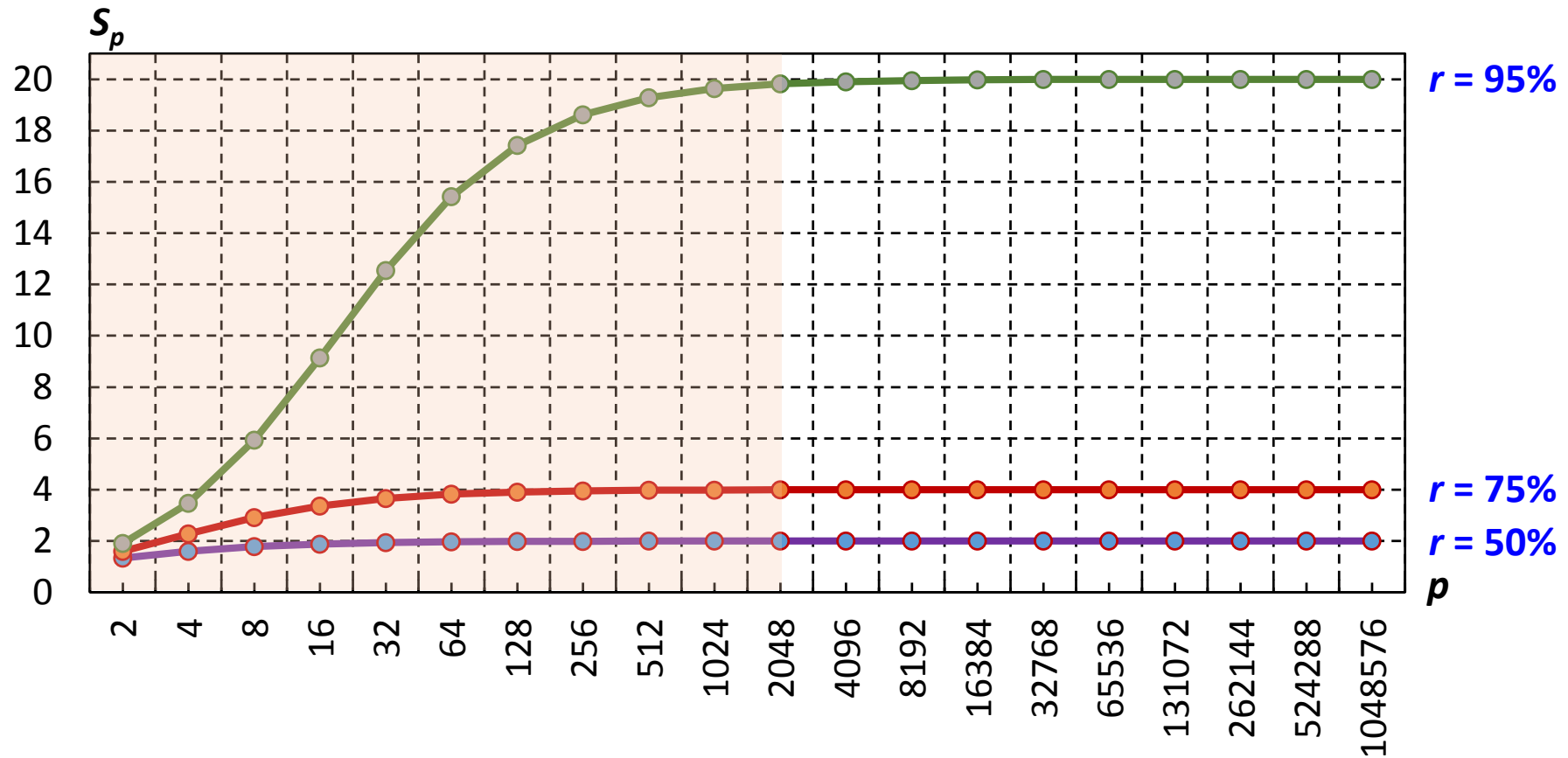
Максимальное ускорение S_p программы на p процессорах равняется

$$S_p = \frac{1}{(1 - r) + \frac{r}{p}}$$
$$S_\infty = \lim_{p \rightarrow \infty} S_p = \lim_{p \rightarrow \infty} \frac{1}{(1 - r) + \frac{r}{p}} = \frac{1}{1 - r} = \frac{1}{s}$$

- ❑ Amdahl Gene. **Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities** // AFIPS Conference Proceedings, 1967, pp. 483-485, <http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf>



Закон Дж. Амдала (Amdahl's law)



Зависимость коэффициента S_p ускорения
параллельной программы от количества p процессоров

Имеет ли смысл создавать системы с количеством процессоров > 1024 ?

Пример

```
// Последовательная часть: инициализация
x = (int *)calloc(n, sizeof(int));

// Распараллеливаемая часть
do {
    for (i = 0; i < n; i++) {
        x[i] = f(i); // O(1)
    }

    // Проверка сходимости
    done = ... ; // O(1)
} while (!done)
```

- Пусть для определенности, цикл **do** завершается после k итераций
- Цикл **for** можно эффективно распараллелить
- $T_{seq}(n) = n + k + kn$
- $T_{par}(n) = n + k + kn/p$
- $s = \frac{n+k}{n+k+kn}, r = \frac{kn}{n+k+kn}$

- Тогда доля последовательного кода при $n \rightarrow \infty$:

$$\lim_{n \rightarrow \infty} s = \lim_{n \rightarrow \infty} \frac{n+k}{n+k+kn} = \frac{1}{1+k}$$

- Ускорение $S_{\infty} = 1/s = 1+k$

Пример

```
// Последовательная часть: инициализация
x = (int *)malloc(n * sizeof(int));

// Распараллеливаемая часть
do {
    for (i = 0; i < n; i++) {
        x[i] = f(i); // O(1)
    }

    // Проверка сходимости
    done = ... ; // O(1)
} while (!done)
```

- Пусть для определенности, цикл **do** завершается после k итераций
- Цикл **for** можно эффективно распараллелить
- $T_{seq}(n) = 1 + k + kn$
- $T_{par}(n) = 1 + k + kn/p$
- $s = \frac{1+k}{1+k+kn}, r = \frac{kn}{1+k+kn}$

Ускорение $S \rightarrow p$, при $n \rightarrow \infty$

Допущения закона Дж. Амдала (Amdahl's law)

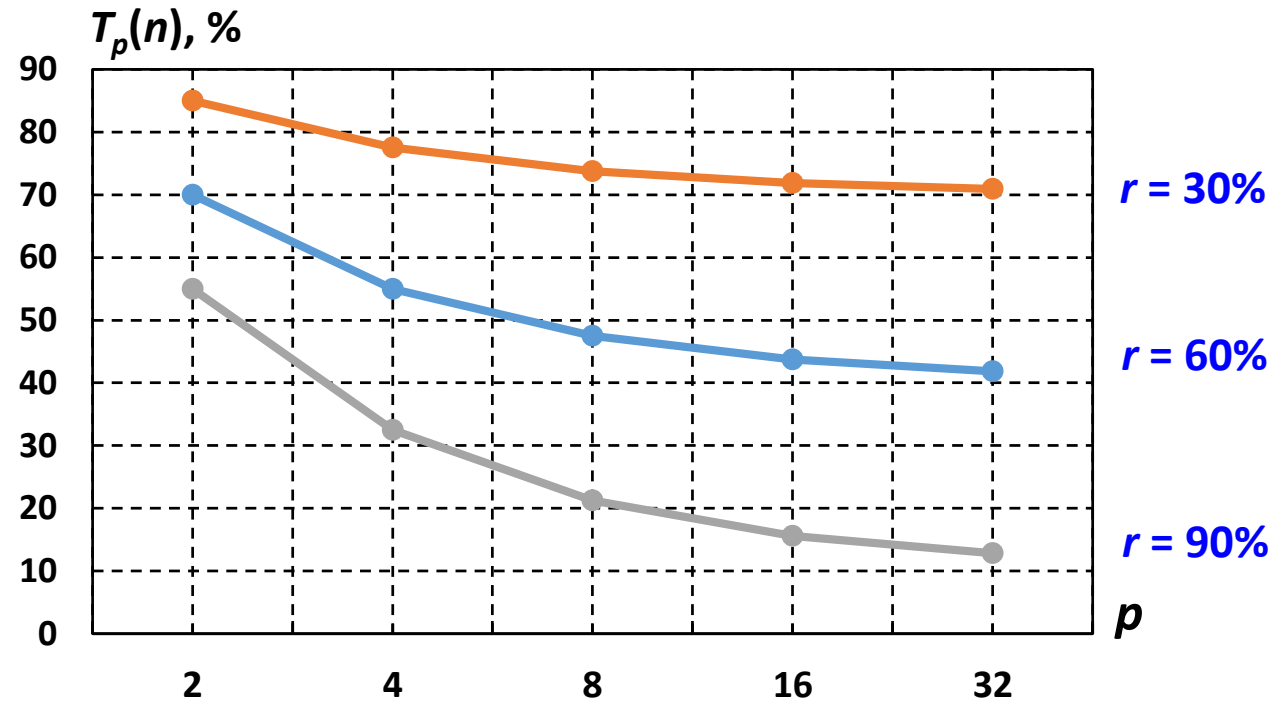
- **Последовательный алгоритм является наиболее оптимальным способом решения задачи**
- Возможны ситуации когда параллельная программа (алгоритм) эффективнее решает задачу (может эффективнее использовать кеш-память, конвейер, SIMD-инструкции, ...)
- **Время выполнения параллельной программы оценивается через время выполнения последовательной**, однако потоки параллельной программы могут выполняться эффективнее

$$T_p(n) = T(n)s + \frac{T(n)}{p}r, \quad \text{на практике возможна ситуация } \frac{T(n)}{p} > T_p(n)$$

- **Ускорение $S_p(n)$ оценивается для фиксированного размера n данных при любых значениях p**
- В реальности при увеличении числа используемых процессоров размер n входных данных также увеличивают, так как может быть доступно больше памяти

Закон Дж. Амдала (Amdahl's law)

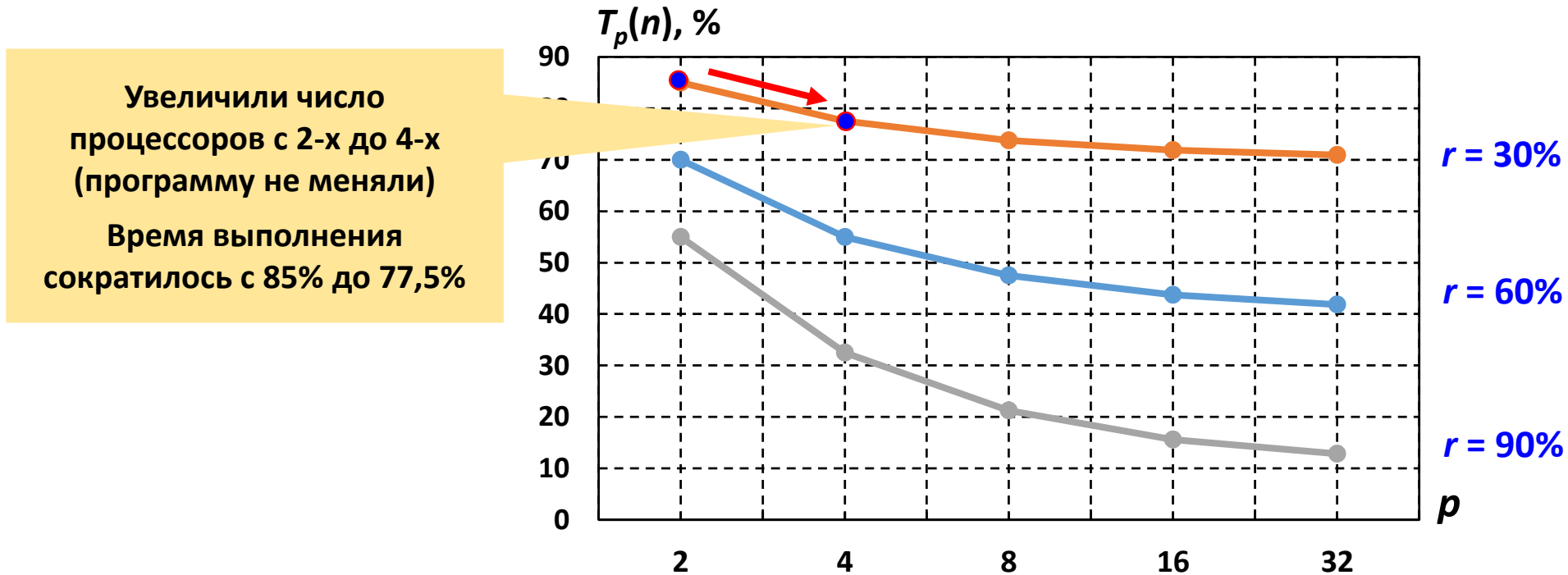
- На что потратить ресурсы – на увеличение доли r параллельной части в программе или увеличение числа процессоров, на которых запускается программа?



Зависимость времени $T_p(n)$ выполнения параллельной программы от количества p процессоров и доли r распараллеленного кода (время в % от времени $T_1(n)$)

Закон Дж. Амдала (Amdahl's law)

- На что потратить ресурсы – на увеличение доли r параллельной части в программе или увеличение числа процессоров, на которых запускается программа?



Зависимость времени $T_p(n)$ выполнения параллельной программы от количества p процессоров и доли r распараллеленного кода (время в % от времени $T_1(n)$)

Закон Дж. Амдала (Amdahl's law)

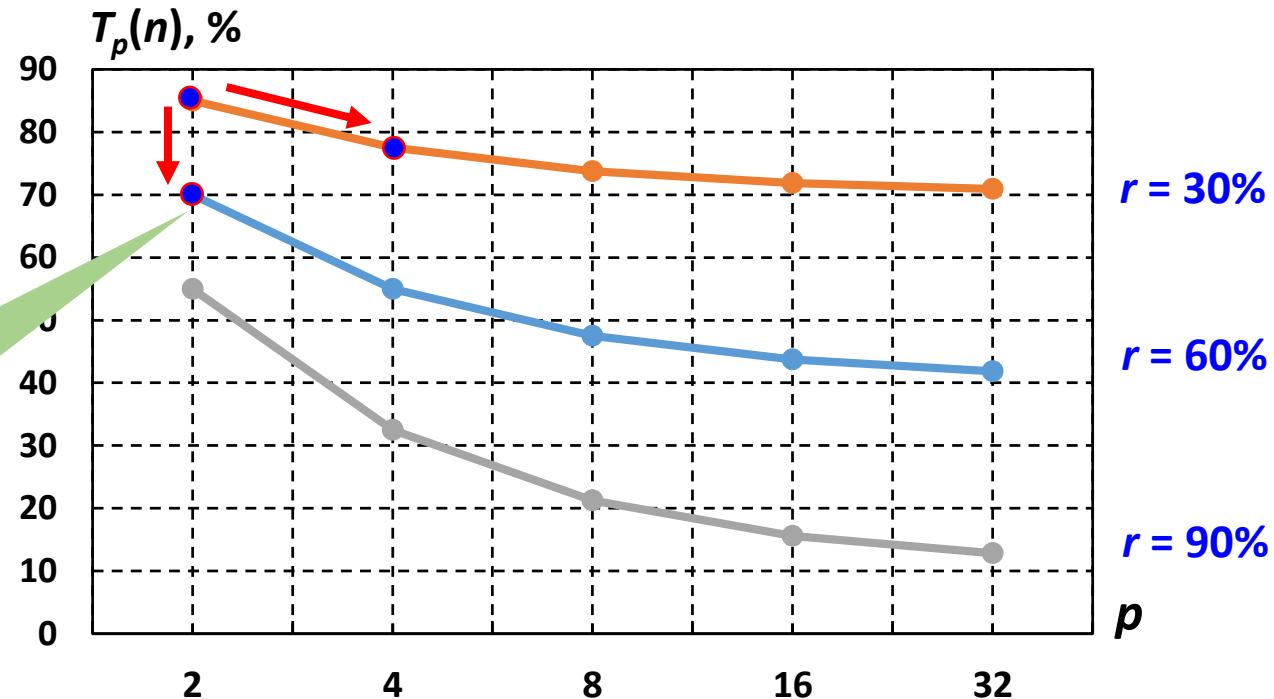
- На что потратить ресурсы – на увеличение доли r параллельной части в программе или увеличение числа процессоров, на которых запускается программа?

Увеличили число процессоров с 2-х до 4-х (программу не меняли)

Время выполнения сократилось с 85% до 77,5%

Увеличим в 2 раза долю параллельного кода

Время выполнения сократилось с 85% до 70%



Зависимость времени $T_p(n)$ выполнения параллельной программы от количества p процессоров и доли r распараллеленного кода (время в % от времени $T_1(n)$)

Закон Густафсона-Барсиса

- Пусть имеется последовательная программа с временем выполнения $T(n)$
- Обозначим $s \in [0, 1]$ – часть параллельной программы, которая выполняется последовательно (purely sequential)
- **Закон Густафсона-Барсиса (Gustafson–Barsis' law) [1]:**

Масштабируемое ускорение S_p программы на p процессорах равняется

$$S_p = p - s(p - 1)$$

- **Обоснование:** пусть a – время последовательной части, b – время параллельной части

$$T_p(n) = a + b, \quad T(n) = a + pb$$

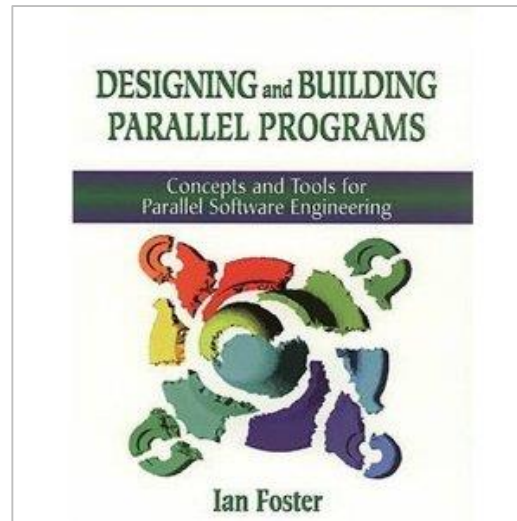
$$s = a/(a + b), \quad S_p(n) = s + p(1 - s) = p - s(p - 1)$$

- **Время выполнения последовательной программы выражается через время выполнения параллельной**
- **Reevaluating Amdahl's Law**, John L. Gustafson, Communications of the ACM 31(5), 1988. pp. 532-533 // <http://www.scl.ameslab.gov/Publications/Gus/AmdahlsLaw/Amdahls.html>

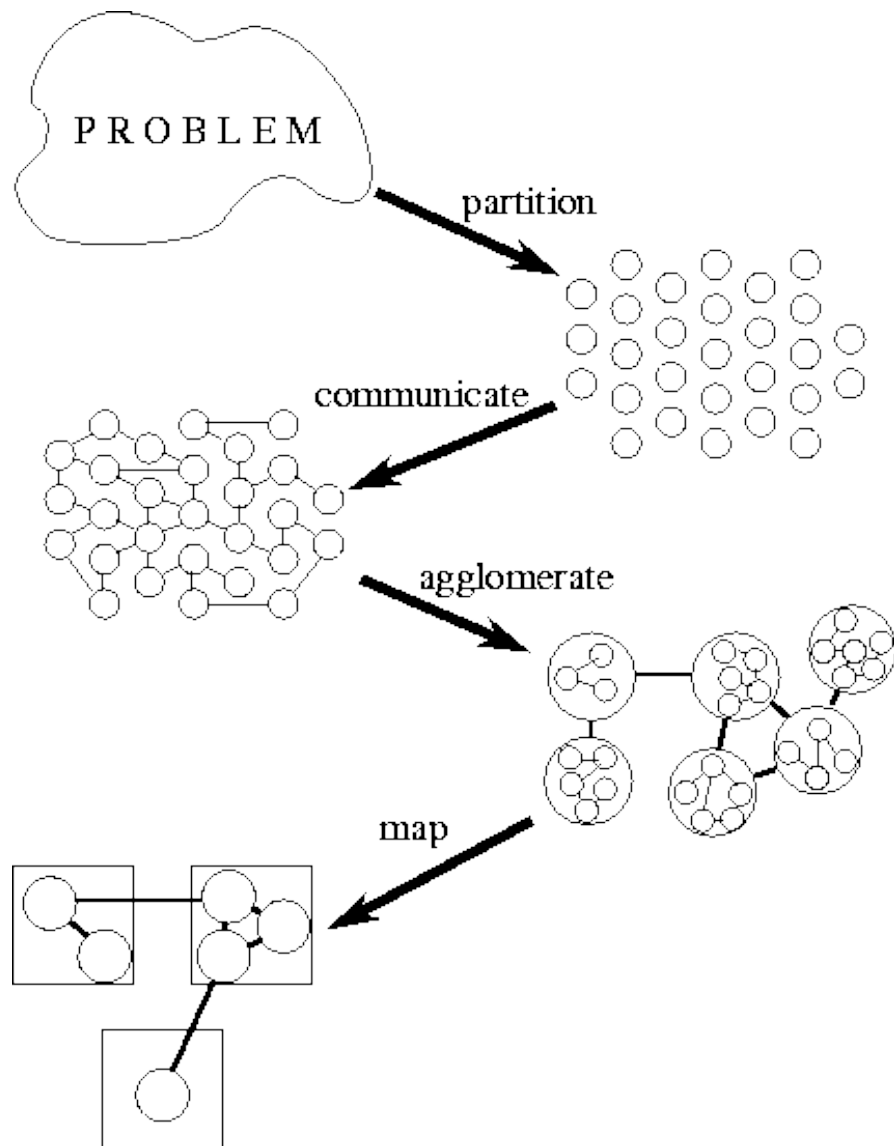
Принципы разработки параллельных алгоритмов

Методология PCAM

- **Методология PCAM** описывает общий подход к процессу разработки параллельного алгоритма для решения заданной задачи
- **PCAM** = Partitioning, Communication, Agglomeration, Mapping
- Foster I. **Designing and Building Parallel Programs: Concepts and Tools for Software Engineering**. Reading, MA: Addison-Wesley, 1995 // <http://www.mcs.anl.gov/~itf/dbpp/>



Методология РСАМ



- **Декомпозиция** (Partition) вычислений и данных на параллельные подзадачи (архитектура ВС игнорируется)
- **Анализ зависимостей и разработка алгоритма взаимодействия** (Communicate) параллельных подзадач
- Выбор целевой вычислительной системы (класса)
- Масштабирование подзадач (Agglomerate) с учетом архитектуры выбранной ВС
- Распределение подзадач (Map) между процессорами (статическое или динамическое)

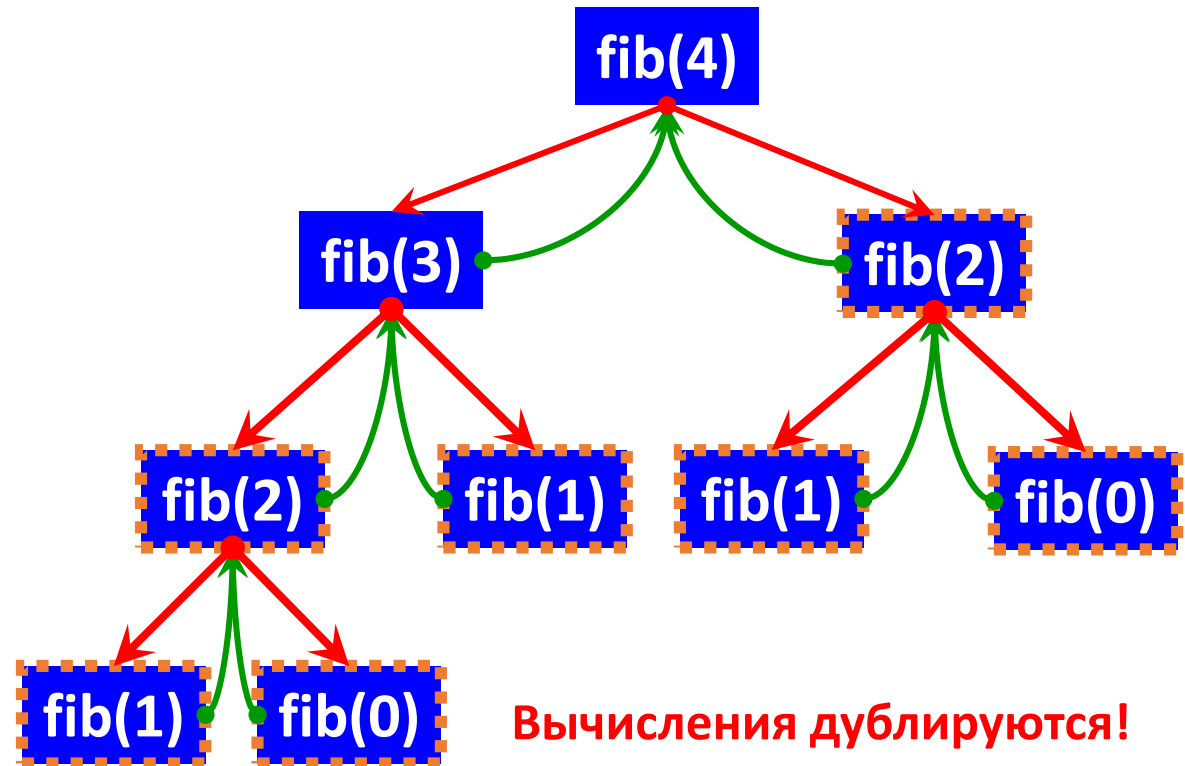
Декомпозиция на подзадачи

- **Выявление возможностей для параллельного выполнения**
- Разбиение задачи на подзадачи минимального размера – *fine-grained decomposition* (мелокзернистая декомпозиция)
- **Виды декомпозиции**
 - **По данным (Domain decomposition)** – распределение данных по подзадачам
 - **Функциональная декомпозиция (Functional decomposition)** – распределение вычислений по подзадачам
- Необходимо избегать дублирования вычислений и данных

Функциональная декомпозиция

```
function fib(int n)
  if n < 2 then
    return n
  x = fork task fib(n - 1)
  y = fork task fib(n - 2)
  join threadX
  join threadY
  return x + y
end function
```

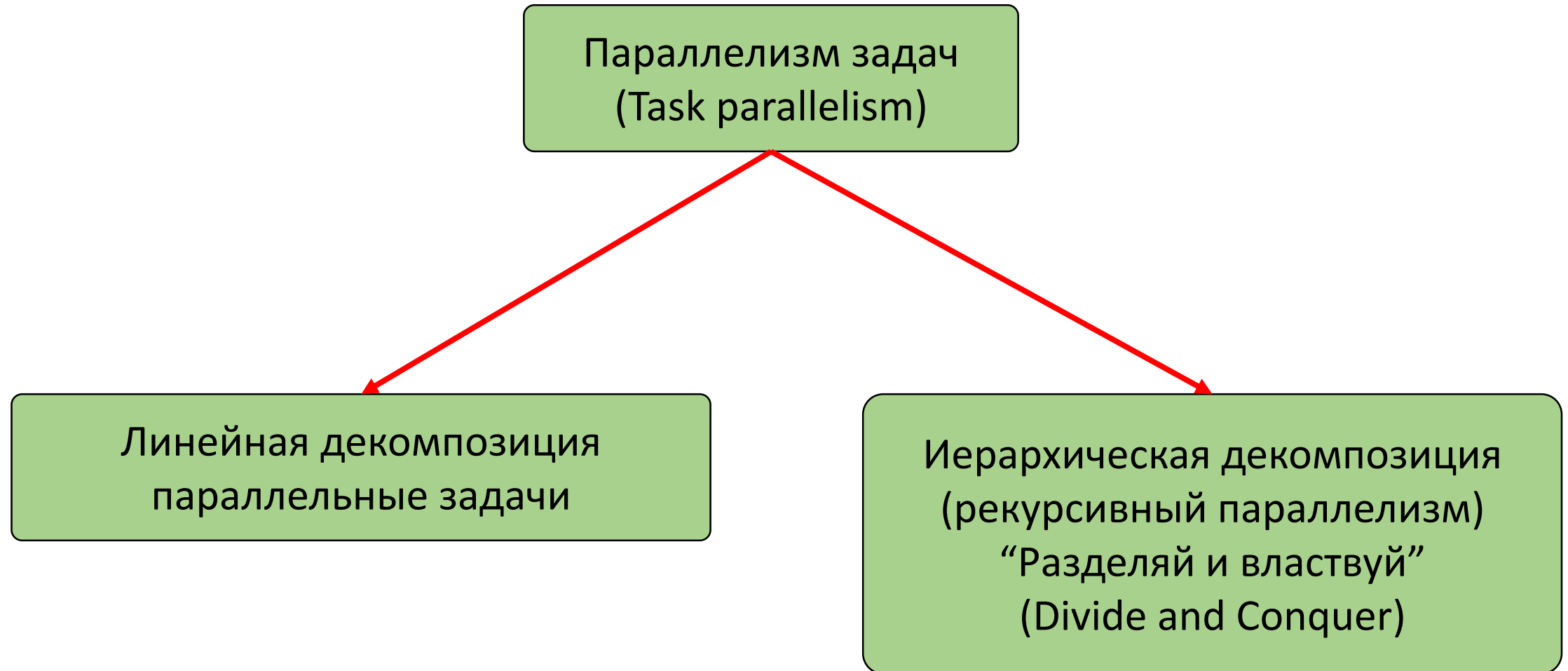
- Параллельное вычисление n -ого члена последовательности Фибоначчи
- Функциональная декомпозиция: каждый рекурсивный вызов – это отдельная подзадача



Выбор структуры алгоритма

- **Существуют типовые структуры (паттерны) параллельных алгоритмов**
- Mattson T., Sanders B., Massingill B. **Patterns for Parallel Programming.** – Addison-Wesley, 2004
- Krste Asanovic, Ras Bodik, Bryan Christopher et. al. **The Landscape of Parallel Computing Research: A View from Berkeley** // <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf>
- **Dwarf Mine** // <http://view.eecs.berkeley.edu/wiki/Dwarfs>

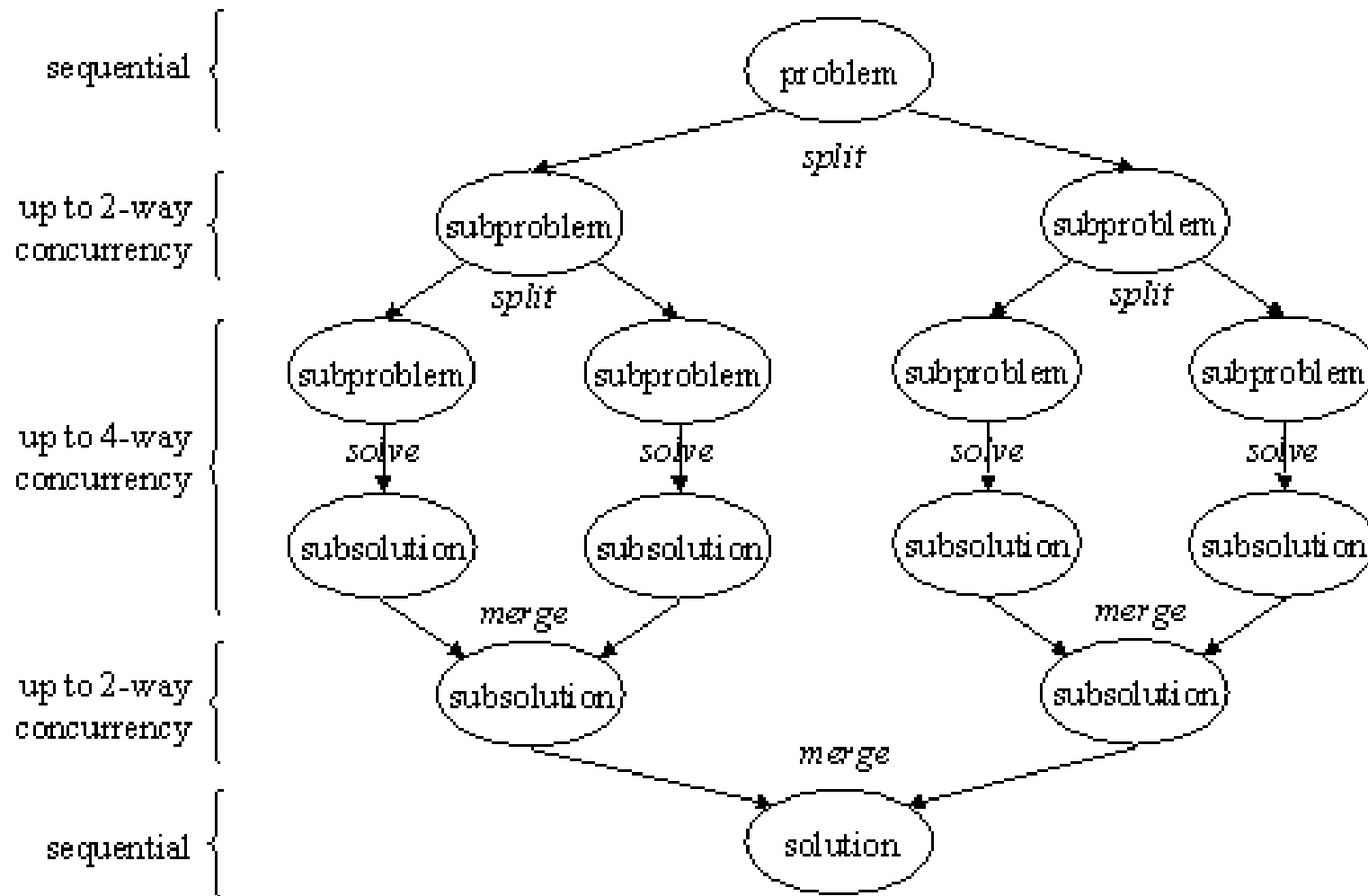
Функциональная декомпозиция



Параллелизм задач (Task parallelism)

- Многовариантный счет, методы Монте-Карло, рендеринг графических сцен
 - Большое количество параллельных подзадач, между задачами нет зависимостей по данным (embarrassingly parallel)
- Молекулярная динамика (система из n взаимодействующих атомов)
 - Параллельное вычисление сил, действующих на атом
- Метод «ветвей и границ» (branch and bound)
 - Обход и разбиение множества решений в соответствии с правилами отсева и ветвления
 - Динамическое порождение заданий

Рекурсивный параллелизм (разделяй и властвуй)

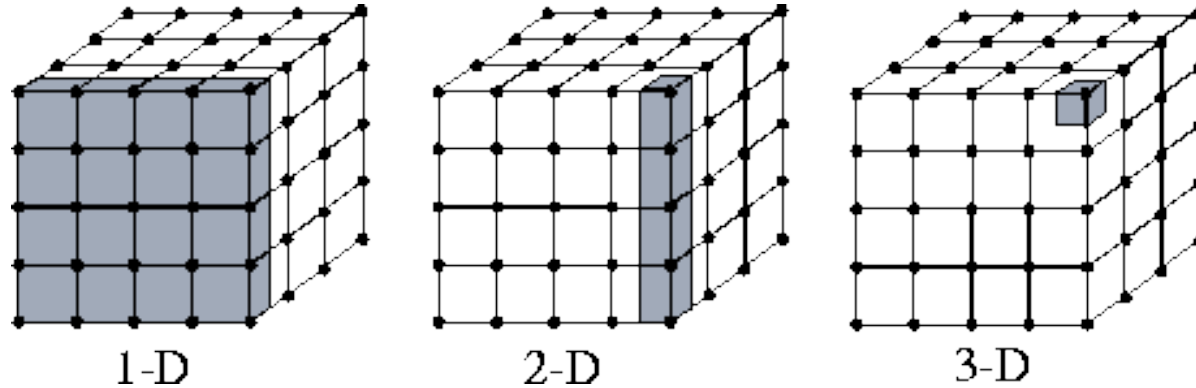


- Parallel Merge Sort
- Parallel Quick Sort

Рекурсивный параллелизм (разделяй и властвуй)

- Степень параллелизма изменяется в ходе выполнения алгоритма
- Операции Split и Merge могут стать узким местом (выполняются последовательно, см. закон Амдала)
- Задания порождаются динамически (балансировка загрузки потоков)
- Очень большое количество заданий может привести к значительным накладным расходам

Геометрическая декомпозиция (Domain decomposition)



- **Данные задачи разбиваются на области** (желательно равного размера)
- С каждой областью данных ассоциируются алгоритм её обработки
- **Вычисления локализованы внутри области?**
 - Да: независимые подзадачи
 - Нет: требуется разделение данных между областями

Геометрическая декомпозиция (Domain decomposition)

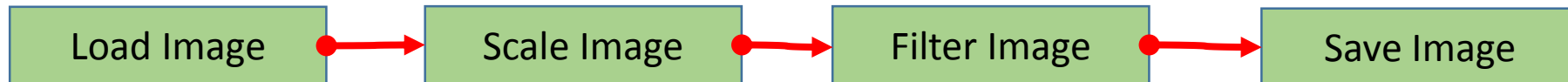
- **Декомпозиция структуры данных на области**
 - Размер подзадач обычно подбирается эмпирически
 - Форма области влияет на накладные расходы (отношение объема к площади поверхности)
 - Дублирование соседних точек
- **Реализация обмена данными**
 - Перед операцией обновления
 - Параллельно с операцией обновления

Рекурсивная декомпозиция

- Алгоритм работает с рекурсивной структурой данных (список, дерево, граф)
- Часто кажется, что единственный способ решения – последовательный обход структуры
- Однако иногда возможно перестроить алгоритм так, что операции над отдельными элементами можно выполнять одновременно
- Vivek Sarkar. **Parallel Graph Algorithms** // <http://www.cs.rice.edu/~vs3/comp422/lecture-notes/comp422-lec24-s08-v2.pdf>

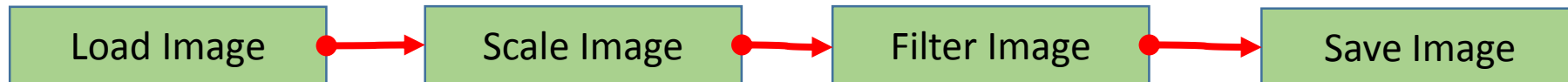
Конвейерная обработка (Pipeline)

- Вычисления производятся над набором элементов данных, каждый из которых проходит несколько стадий обработки – этапы/блоки конвейера
- Регулярный, односторонний, стабильный поток данных
- Подзадачи – применение операции "стадия N" к каждому элементу данных
- Примеры
 - Конвейерная обработка команд процессором
 - Обработка сигналов, фильтры, графика



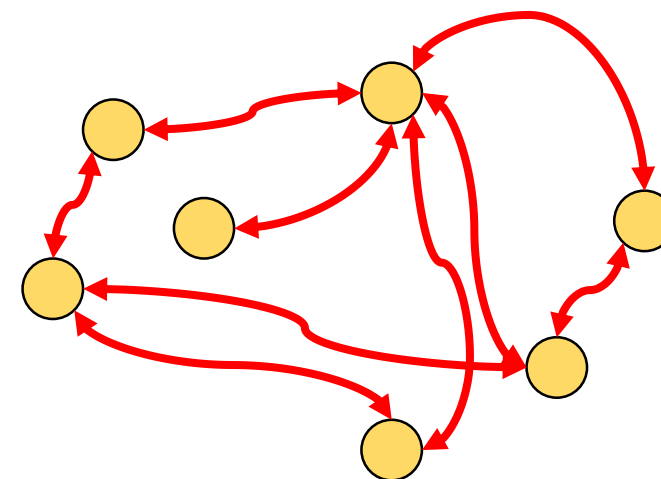
Конвейерная обработка (Pipeline)

- Параллелизм ограничен числом стадий
- В идеале времена работы каждой стадии должны быть одинаковыми
 - Самая медленная стадия становится узким местом
 - Комбинирование и декомпозиция стадий
 - Распараллеливание медленной стадии
- Времена заполнения и опустошения конвейера



Координация на основе событий

- Декомпозиция на слабосвязанные компоненты, взаимодействующие нерегулярным образом
- Двусторонние потоки данных
- Нерегулярные, непредсказуемые взаимодействия
- Высокий риск возникновения взаимной блокировки
- **Примеры**
 - ❑ Моделирование с дискретными событиями
 - ❑ Координация между заданиями в других шаблонах
 - ❑ Распределенные системы

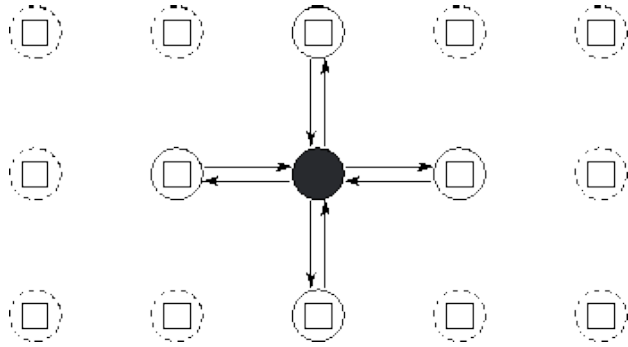


**Граф взаимодействий подзадач
может быть недетерминированным**

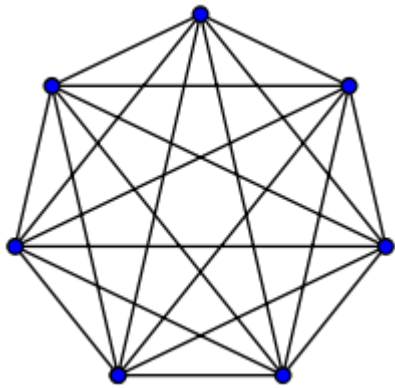
Взаимодействия между подзадачами

- Локальные и глобальные
- Структурированные и неструктурированные
- Статические и динамические
- Синхронные и асинхронные

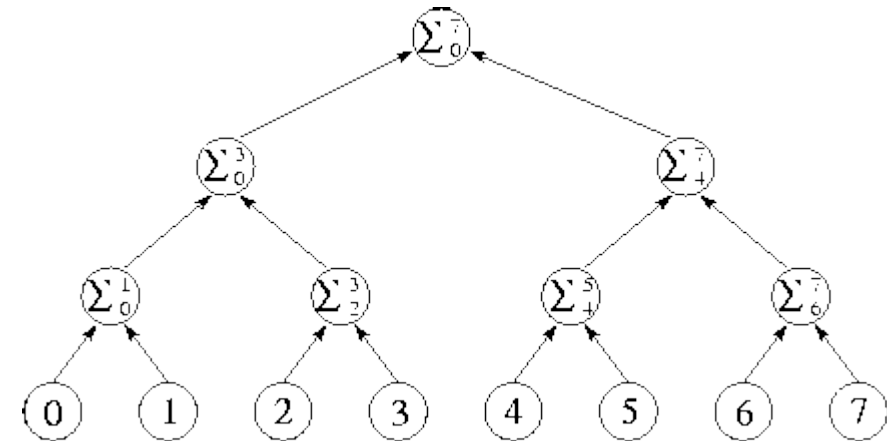
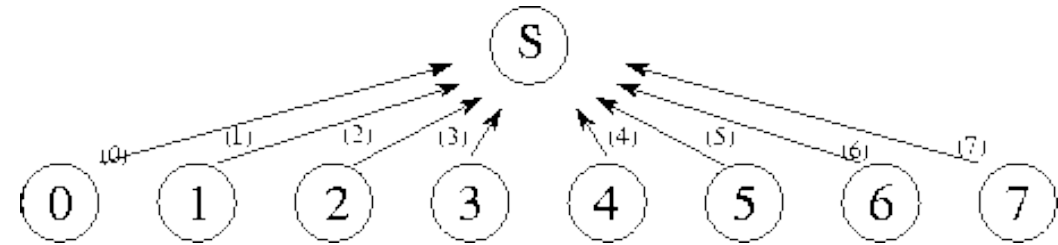
Взаимодействия между подзадачами



Локальные взаимодействия



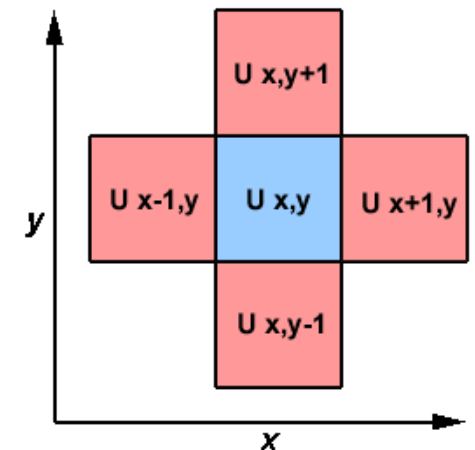
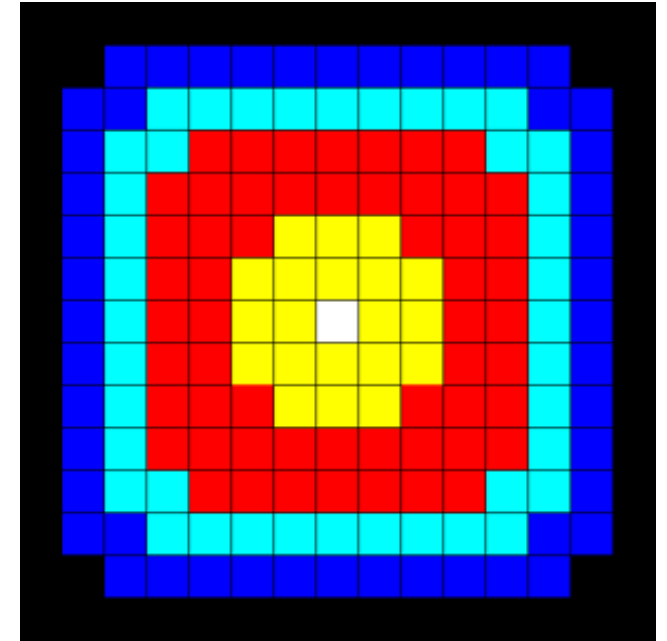
Коллективные операции
All-to-all



Коллективные (глобальные) операции
Reduction - сборка

Heat 2D (serial code)

- Уравнение теплопроводности описывает изменение температуры в заданной области с течением времени
- Приближенное решение можно найти методом конечных разностей
- Область покрывается сеткой
- Производные аппроксимируются конечными разностями
- Известна температура на границе области в начальный момент времени

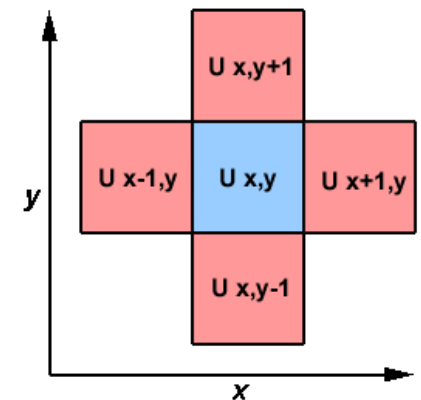
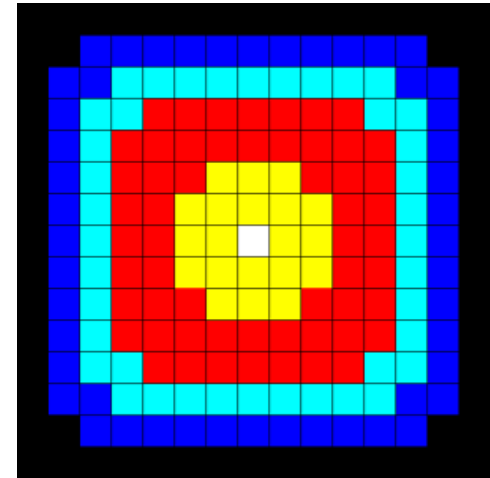


[*] Blaise Barney. Introduction to Parallel Computing (LLNL)

Heat 2D (serial code)

- Температура хранится в двумерном массиве – расчетная область

```
do iy = 2, ny - 1
  do ix = 2, nx - 1
    u2(ix, iy) = u1(ix, iy) +
      cx * (u1(ix + 1, iy) +
        u1(ix - 1, iy) - 2.0 * u1(ix, iy)) +
      cy * (u1(ix, iy + 1) +
        u1(ix, iy - 1) - 2.0 * u1(ix, iy))
  end do
end do
```

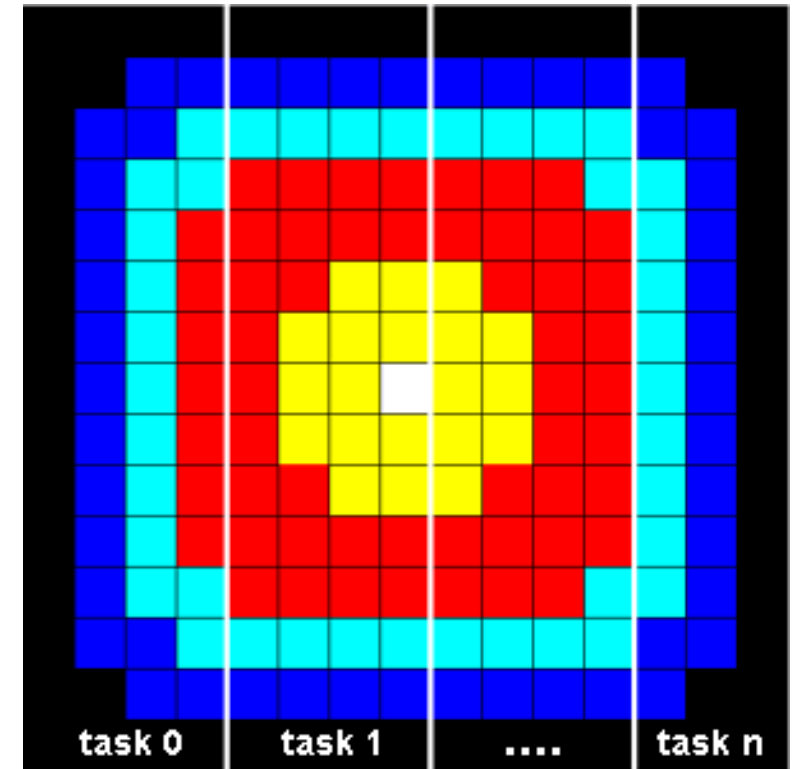
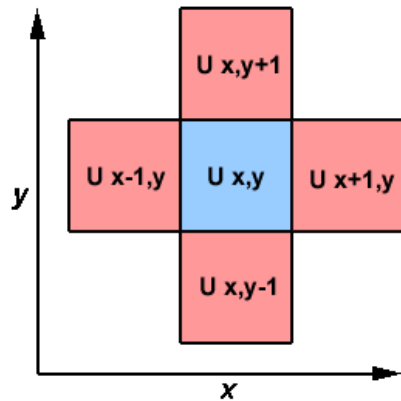


[*] Blaise Barney. Introduction to Parallel Computing (LLNL)

Heat 2D (serial code)

- Каждый процесс будет обрабатывать свою часть области – **1D domain decomposition**

```
for t = 1 to nsteps  
  1. Update time  
  2. Send neighbors my border  
  3. Receive from neighbors  
  4. Update my cells  
end for
```



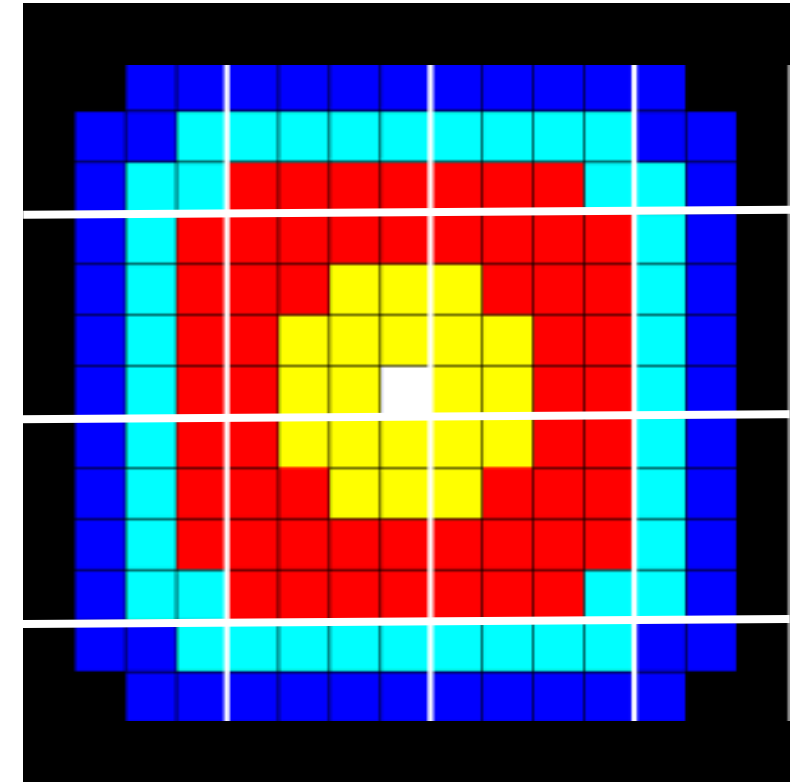
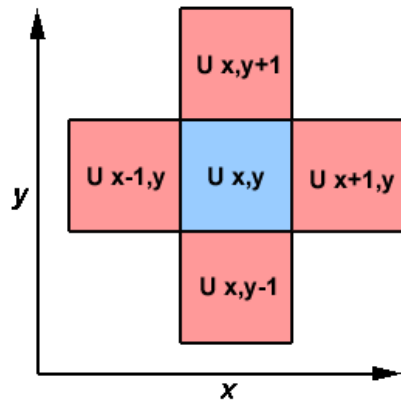
Расчетная область разбита на
вертикальные полосы –
массив распределен

[*] Blaise Barney. Introduction to Parallel Computing (LLNL)

Heat 2D (serial code)

- Каждый процесс будет обрабатывать свою часть области – **2D domain decomposition**

```
for t = 1 to nsteps  
  1. Update time  
  2. Send neighbors my border  
  3. Receive from neighbors  
  4. Update my cells  
end for
```



Расчетная область разбита на
прямоугольные области (2D)

Сколько надо выполнить обменов
на каждом шаге?

[*] Blaise Barney. Introduction to Parallel Computing (LLNL)

- Foster I. **Designing and Building Parallel Programs: Concepts and Tools for Software Engineering** – <http://www.mcs.anl.gov/~itf/dbpp/>
- Herb Sutter. **Welcome to the Jungle** – <http://herbsutter.com/welcome-to-the-jungle/>