ОПТИМИЗАЦИЯ КОЛИЧЕСТВА ПРОЦЕССОРОВ ДЛЯ ЭФФЕКТИВНОГО ИСПОЛНЕНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ

О.А. Юфрякова

В настоящее время принято разделять вычислительную сложность параллельного алгоритма (без учета времени на коммуникации) и коммуникативную сложность [2]. При этом постоянно делаются оговорки о том, что количество процессоров (или узлов кластера) разумно ограничить некоторым числом, определяемым производительностью сети коммуникаций и законами Амдала [1,2,3]. Но оценки этого «разумного» числа процессоров не приводятся. Такие оценки вычисленные для различных архитектур были бы несомненно полезными для планирования эффективного использования ресурсов вычислительной системы (ВС). Оценки, приведенные в работе [5], имеют чисто теоретический интерес и не содержат явных выражений через объем входных данных и существенные характеристики ВС. В этой работе предлагается общий подход, позволяющий оценить оптимальное число процессоров ВС, и приводится несколько конкретных оценок.

Эффективность реальных ВС на широких классах параллельных алгоритмов обеспечивается гармоничным взаимодействием вычислительных процессов и процессов обменов данными.

Введем основные понятия и обозначения.

Пусть t_c – среднее время выполнения локальных вычислительных операций таких, как сложение, умножение, обращение к локальной памяти процессора. Характерный масштаб t_c – несколько десятков тактов работы процессора.

Пусть t_s – среднее время пересылки элементарного пакета данных между соседними процессорами по сети коммуникаций. Оно складывается из времени прохождения сигнала по кабелю, латентности и задержек при передаче данных при превышении пропускной способности.

Например, если тактовая частота процессора 3 ГГц, т.е. $3 \cdot 10^9$ тактов в сек, то 1 такт = $0{,}33 \cdot 10^{-9}$ сек. Скорость света 300000 км/сек = $3 \cdot 10^8$ м/с. За один такт свет пройдет 1/10 м = 10 см. Таким образом каждый метр кабеля дает принципиально неустранимую задержку в 10 тактов. С ростом тактовой частоты эта задержка в тактах только увеличивается.

Если латентность в сети коммуникаций ВС, как у Cray T3E-1200E, равна 1 мкс = 10^{-6} сек = 3300 тактов, то характерное время выполнения операций пересылки данных на два порядка превосходит время выполнения вычислительных операций. В сетях Ethernet латентность в 10-100 раз выше чем у Cray.

Мы видим, что именно операции пересылки данных являются наиболее медленными даже для специализированных суперкомпьютеров и являются узким местом всех параллельных ВС. Поэтому распределение данных должно быть по возможности крупноблочным и по вычислениям и по пересылке. Как мы увидим в дальнейшем, отношение t_s / t_c является важнейшей характеристикой ВС, показывающей сбалансированность ее вычислительной и коммуникационной мощностей. Это отношение показывает, сколько в среднем вычислительных операций должен выполнить процессор на одну операцию пересылки данных соседним процессорам.

Вычислительные операции, выполняемые на одном процессоре, и операции по пересылке данных между соседними в топологии ВС процессорами мы будем называть *локальными*.

В любом параллельном алгоритме присутствуют также *глобальные* операции – создание виртуальной ВС, начальная рассылка данных, сбор результатов вычислений с процессоров (broadcast, scatter, reduce, gather и т.д.). При оценке сложности о них обычно умалчивается или предполагается, что они уже выполнены. При этом говорится, что выполнена предварительная подготовка данных или, что данные уже распределены по памяти процессоров. На самом деле это существенная часть параллельного алгоритма, использующая массовые обмены данными. Поэтому при оценке сложности параллельных алгоритмов следует совместно рассматривать вычислительные и коммуникационные аспекты. Только на этом пути можно получить оценки для оптимального числа процессоров, минимизирующее время исполнения параллельного алгоритма как функцию от объема исходных данных n для реальных ВС.

Сначала рассмотрим идеальный, с точки зрения законов Амдала, параллельный алгоритм A, исполняемый на однородной ВС.

$C_A(n)$ – временная вычислительная сложность A, как ее понимают в классической теории сложности вычислений, т.е. асимптотика при $n \to \infty$ суммарного количества элементарных вычислительных операций необходимых для последовательного исполнения алгоритма A на данных объемом n в худшем случае.

$S^l_A(n)$ – локальная коммуникационная сложность A, т.е. асимптотика (при $n \to \infty$) суммарного количества локальных элементарных операций пересылки данных между соседними процессорами, необходимых для исполнения параллельного алгоритма A при заданной топологии ОВС на задаче объемом n в худшем случае.

$S^g_A(n)$ – коммуникационная сложность глобальных операций алгоритма A – количество элементарных операций пересылки данных на каждый один процессор при инициализации виртуальной ВС, начальном

распределении данных, глобальной синхронизации и так далее. Общее время исполнения этих операций, очевидно, пропорционально количеству *P* элементарных машин.

Тогда время исполнения всего параллельного алгоритма *A* на ОВС из *P* процессоров будет равно:

$$T_A(n,p) = \frac{C_A(n)\cdot t_c + S_A^l(n)\cdot t_s}{P} + S_A^g(n)\cdot P\cdot t_s$$

Оценим количество процессоров *P*, при котором время исполнения алгоритма будет минимальным. Как функция от *P* это выражение изменяется гладко, без осцилляций и имеет единственный минимум при *P>0*. В окрестности минимума функция меняется медленно при относительно малых значениях *P*. Поэтому этот минимум найдем методами классического анализа функций.

Утверждение. Время исполнения идеального параллельного алгоритма *A* на ОВС из *P* процессоров будет наименьшим при

$$P_{опт} \approx \left\lceil \left(\frac{C_A(n)\cdot t_c + S_A^l(n)\cdot t_s}{S_A^g(n)\cdot t_s} \right)^{\frac{1}{2}} \right\rceil$$

Замечание 1. *P* зависит от графа коммутаций и не может быть произвольным натуральным числом. Например, для гиперкуба $P = 2^N$, для *2D*-тора *P= MxN* Поэтому *P* надо брать ближайшим справа возможным числом к приведенной оценке.

Замечание 2. Для увеличения масштабируемости необходимо уменьшать t_s и $S^g_A(n)$, выражая глобальные коммуникационные операции через локальные, тем самым распараллеливая глобальные операции.

При определении эффективности параллельных алгоритмов сравнивают времена исполнения $T_A(n,1)$ и $T_A(n,P)$. На одном процессоре нет операций коммутации, значит, $S^g_A(n)=0$ и $S^l_A(n)=0$. Поэтому «полезными» считают только вычислительные операции. Разобьем $T_A(n,P)$ на две части – «полезную» и коммуникационную:

$$T_A(n,P) = \underbrace{\frac{C_A(n)\cdot t_c}{P}}_{\text{«полезная»}} + \underbrace{\frac{S_A^l(n)\cdot t_s}{P} + S_A^g(n)\cdot P\cdot t_s}_{\text{коммуникационная}}$$

В силу выше сказанного, естественным кажется требование:

$$\frac{C_A(n)}{P} \geq \frac{S_A^l(n)}{P} + S_A^g(n)\cdot P$$

– «полезные» операции должны составлять не меньшую часть, чем «бесполезные». Это требование также приводит к ограничению для числа *P*

$$P \leq \sqrt{\frac{C_A(n)\cdot t_c - S_A^l(n)\cdot t_s}{S_A^g(n)\cdot t_s}},$$

при условии, что $C_A(n)\cdot t_c \geq S_A^l(n)\cdot t_s$. Последнее неравенство не зависит от *P* и ограничивает разумное количество локальных пересылок данных $S_A^l(n) \leq C_A(n)\frac{t_c}{t_s}$ в «приличных» параллельных алгоритмах.

По сути приведенные ограничения являются еще одним дополнительным коммуникационным аспектом законов типа Амдала. Было бы интересно проверить эти ограничения на конкретных параллельных алгоритмах в различных вычислительных средах.

Для неидеальных алгоритмов, имеющих нераспараллеливаемые цепочки вычислительных и коммуникационных операций, выражения $C_A(n)$ и $S_A^l(n)$ сами могут зависеть от *P*. Поэтому для них оценки оптимального числа процессоров несколько изменяются. Но сам подход остается прежним. Дело в том, что при любой организации рассылки данных всегда будет хотя бы одна глобальная операция коммуникации. Например, всегда надо сообщать процессорам логические номера их соседей в зависимости от топологии виртуальной ВС, сообщать о начале и окончании процесса вычислений или других видах глобальной синхронизации. В лучшем случае сложность этих операций не зависит от объема исходных данных и пропорциональна числу процессоров *P*. Коэффициент пропорциональности обозначим буквой *K*. Он имеет простой смысл – количество элементарных пакетов данных, полученных и посланных каждым процессором. Таким образом, в выражение для времени исполнения алгоритма $T_A(n,P)$ обязательно будет входить слагаемое вида KPt_s. Рассмотрим ряд оценок для таких алгоритмов.

1. Сложение n чисел на P процессорах с рассылкой данных с одного процессора на все, сбором результатов со всех процессоров на одном и суммированием результатов на этом процессоре. Время исполнения этого алгоритма таково:

$$T_A(n,P) = \frac{nt_c}{P} + Pt_c + \frac{nP}{P}t_s + KPt_s = \frac{nt_c}{P} + nt_s + (t_c + Kt_s)P .$$

Мы видим, что в формуле появилось слагаемое, независящее от P. Тем самым оно не влияет на оценку оптимального числа процессоров $P_{опт}$. Кроме того, цепочка сложений результатов на одном процессоре дала дополнительный вклад в коэффициент t_c при P.

Поскольку приведенная выше оценка для $P_{опт}$ зависела только от коэффициентов при P^{-1} и P, то можно воспользоваться ей подставив вместо $C_A(n)$, $S_A^g(n)$ и $S_A^l(n)$ эти коэффициенты.

$$P_{опт} \approx \left\lceil \left(\frac{nt_c}{t_c + Kt_s} \right)^{\frac{1}{2}} \right\rceil = \left\lceil \sqrt{n} \left(\frac{t_c}{t_c + Kt_s} \right)^{\frac{1}{2}} \right\rceil$$

Если $n=10^6$, $t_c=1$, $t_s=1000$, $K=5$, то $P_{опт} \approx \left\lceil \frac{1000}{\sqrt{5001}} \right\rceil \approx 15$. Эта оценка справедлива для сверхбыстрых коммуникационных архитектур (с латентностью в три раза меньше, чем у Cray). Для кластеров $t_s=30000$, $K=15$, значит, $P_{опт} \approx \left\lceil \frac{1000}{\sqrt{450001}} \right\rceil = 2$.

Аналогично получаем оценки для умножения, но для умножения $t_c=10$. Приведенные оценки показывают, что каскадные схемы сложения и умножения с более чем одним каскадом на практике пока не актуальны. Но полезно асинхронное исполнение этого алгоритма с одним выделенным процессором, который занимается исключительно рассылкой и сбором данных а также выполняет суммирование полученных результатов.

2. Скалярное произведение векторов длины n с рассылкой $2n/P$ чисел, сбором результатов и суммированием их на одном процессоре:

$$T_A(n,P) = \frac{2n}{P}t_c + Pt_c + \frac{2nP}{P}t_s + KPt_s = \frac{2nt_c}{P} + 2nt_s + (t_c + Kt_s)P$$

$$P_{опт} \approx \left\lceil \sqrt{n} \left(\frac{2t_c}{t_c + Kt_s} \right)^{\frac{1}{2}} \right\rceil$$

Для суперкомпьютеров $n=10^6$, $t_c=10$, $t_s=1000$, $K=5$, $P_{опт} \approx \left\lceil 1000 \left(\frac{20}{5010} \right)^{\frac{1}{2}} \right\rceil = 64$.

Для кластеров $n=10^6$, $t_c=10$, $t_s=30000$, $K=15$, $P_{опт} \approx \left\lceil 1000 \left(\frac{20}{450010} \right)^{\frac{1}{2}} \right\rceil = 7.$

3. Скалярное произведение векторов длины n с рассылкой $2n$ чисел по всем компьютерам, сбором результатов и суммированием их на одном процессоре:

$$T_A(n,P) = \frac{2n}{P}t_c + Pt_c + 2nPP_s + KPt_s = \frac{2nt_c}{P} + (t_c + (2n+K)t_s)P$$

$$P_{опт} \approx \left\lceil \left(\frac{2nt_c}{(2n+K)t_s + t_c} \right)^{\frac{1}{2}} \right\rceil \approx \left\lceil \sqrt{\frac{2nt_c}{2nt_s}} \right\rceil = 1.$$

Такой алгоритм лучше исполнять на 1 процессоре. Разумеется, этот и следующий алгоритм приведены только для иллюстрации того, как влияют избыточные пересылки на распараллеливаемость алгоритмов на реальных ВС.

4. Умножение матриц $n\times n$ с рассылкой по P процессорам обеих матриц-сомножителей, перемножением горизонтальных и вертикальных полос матриц размерами $n\times\frac{n}{P}$ и сбором результатов:

$$T_A(n,P)=\frac{2n^3}{P}t_c+2n^2Pt_s+\frac{n^2P}{P^2}t_s+KPt_s=\frac{n^3t_c+n^2t_c}{P}+(2n^2+K)Pt_s$$

$$P_{опт}\approx\left\lceil\sqrt{\frac{2n^3t_c+n^2t_s}{(2n^2+K)t_s}}\right\rceil\approx\left\lceil n^{0,5}\cdot\sqrt{\frac{t_c}{t_s}}\right\rceil$$

5. Умножение матриц $n\times n$ с рассылкой по P процессорам горизонтальной полосы $\frac{n}{P}\times n$ левой матрицы-сомножителя и вертикальной полосы $n\times\frac{n}{P}$ правой матрицы-сомножителя и сбором результатов:

$$T_A(n,P)=\frac{2n^3}{P}t_c+\frac{2n^2P}{P}t_s+\frac{n^2P}{P^2}t_s+KPt_s=\frac{2n^3t_c+n^2t_s}{P}+2n^2t_s+\cdot KPt_s$$

$$P_{опт}\approx\left\lceil\sqrt{\frac{2n^3t_c+n^2t_s}{Kt_s}}\right\rceil=\left\lceil n\cdot\sqrt{\frac{2nt_c+t_s}{Kt_s}}\right\rceil\approx\left\lceil n^{1,5}\cdot\sqrt{\frac{2t_c}{Kt_s}}\right\rceil$$

Для суперкомпьютеров: $n=10^4$, $t_c=10$, $t_s=1000$, $K=5$, $P_{опт}\approx\left\lceil 10^6\cdot\left(\frac{20}{5010}\right)^{\frac{1}{2}}\right\rceil=63183.$

Для кластеров: $n=10^4$, $t_c=10$, $t_s=30000$, $K=15$, $P_{опт}\approx\left\lceil 10^6\cdot\left(\frac{20}{450010}\right)^{\frac{1}{2}}\right\rceil=6667.$

6. Блочное умножение матриц $n\times n$ на двумерном торе с рассылкой по P процессорам двух квадратных подблоков размерами $\frac{n}{\sqrt{P}}\times\frac{n}{\sqrt{P}}$ матриц-сомножителей, обмена каждым процессором своими подблоками со всеми процессорами строки тора, в которой стоит этот процессор, и результатами умножения этих подблоков с соседями, суммирования полученных подблоков и сбором результатов (алгоритм Фокса [4]):

$$T_A(n,P)=\frac{2n^3}{P}t_c+\frac{2n^2P}{P}t_s+\frac{n^2P}{P^{\frac{3}{2}}}t_s+KPt_s=2\frac{n^3t_c+n^2t_s}{P}+\frac{n^2}{\sqrt{P}}t_s+KPt_s$$

В этом случае возникает дополнительное слагаемое, содержащее $\sqrt{P}$ в знаменателе, описывающее массовые обмены по строкам. Поэтому оценка из *утверждения* здесь непосредственно неприменима. Но метод по-прежнему работает. Если взять производную по P и приравнять ее к нулю, то получится уравнение следующего вида $CP^2-B\sqrt{P}-A=0$, где коэффициенты A, B, C все положительные, причем A существенно больше B, и B существенно больше C. При этих условиях, уравнение имеет только один положительный корень. Заменив P на X^2, применим известную оценку Маклорена [6, с.223] верхней границы вещественных корней для уравнения $CX^4-BX-A=0.$ Получим $X\le 1+\sqrt[3]{A/C}$. Значит $P\le(1+\sqrt[3]{A/C})^2$. Поскольку время исполнения алгоритма является в большинстве реальных алгоритмов весьма медленно растущей функцией при P больших P_{min}, мы не совершим большой ошибки, если воспользуемся оценкой Маклорена, хотя она довольно груба. В итоге получаем следующую несколько завышенную оценку.

$$P_{опт}\approx\left\lceil\left(1+\sqrt[3]{2\frac{n^3t_c+n^2t_s}{Kt_s}}\right)^2\right\rceil\approx\left\lceil\left(1+n\cdot\sqrt[3]{\frac{2t_c}{Kt_s}}\right)^2\right\rceil\approx\left\lceil n^2\cdot\left(\frac{2t_c}{Kt_s}\right)^{1,5}\right\rceil.$$

Эта оценка заметно лучше оценки алгоритма из пункта 5 при больших n. Алгоритм Фокса, наиболее полно использующий двумерный геометрический параллелизм умножения матриц при вычислениях, выигрывает у более простого алгоритма при исполнении на реальных системах, несмотря на избыточные пересылки данных. Но посмотрим, что даст эта оценка на тех же данных и системах, как в пункте 5.

Для суперкомпьютеров: $n=10^4$, $t_c=10$, $t_s=1000$, $K=5$, $P_{опт} \approx \left\lceil 10^8 \cdot \left(\frac{20}{5010} \right)^{\frac{3}{2}} \right\rceil = 25223.$

Для кластеров: $n=10^4$, $t_c=10$, $t_s=30000$, $K=15$, $P_{опт} \approx \left\lceil 10^8 \cdot \left(\frac{20}{450010} \right)^{\frac{3}{2}} \right\rceil = 30.$

Видим, что массированные обмены данными в алгоритме Фокса все же сказались. На матрицах размеров $10^4 \times 10^4$ он уступает алгоритму из пункта 5.

Приведенные оценки еще не вполне детальны, не учитывают топологии сети и особенностей пакетной передачи данных. С другой стороны они уже позволяют делать нетривиальные прогнозы эффективности тех или иных параллельных алгоритмов на различных реальных системах.

ЛИТЕРАТУРА:

1. В.В. Воеводин, Вл.В. Воеводин Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.
2. В.П. Гергель Высокопроизводительные вычисления для многопроцессорных многоядерных систем: Учебник – М.: Издательство московского университета, 2010. – 544 с. – (Серия «Суперкомпьютерное образование»)
3. В.Д. Корнеев. Параллельное программирование в MPI. – Новосибирск: ИВМиМГ СО РАН, 2002. – 215с.
4. V. Kumar, A. Grama, A. Gupta, G. Karypis Introduction to Parallel Computing. – The Benjamin/Cummings Publishing Company, Inc., 1994.
5. Dimitry P. Bertsekas, John N. Tsitsiklis. Parallel and Distributed Computation. Numerical Methods. – Prentice Hall, Englewood Cliffs, New Jersey, 1989.
6. Д.К. Фаддеев. Лекции по алгебре. – С.Пб.: Лань, 2004. – 416с.