Простые эвристики для планирования распараллеливаемых работ в компьютерных системах с учетом ресурсов

М.Ю. Сахно

Омский филиал Института математики им. С.Л. Соболева СО РАН

Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015-П, https://rscf.ru/project/22-71-10015-П/.

Постановка задачи

- Имеется множество распараллеливаемых работ $\mathcal{J} = \{1, ..., n\}$, m ядер процессора и два ресурса возобновимого типа – шина данных и оперативная память.

Постановка задачи

- Имеется множество распараллеливаемых работ $\mathcal{J} = \{1, ..., n\}$, m ядер процессора и два ресурса возобновимого типа – шина данных и оперативная память.
- Для каждой работы $j \in \mathcal{J}$ задан требуемый процент объема оперативной памяти V_j.

Постановка задачи

- Имеется множество распараллеливаемых работ $\mathcal{J} = \{1, ..., n\}$, m ядер процессора и два ресурса возобновимого типа – шина данных и оперативная память.
- Для каждой работы $j \in \mathcal{J}$ задан требуемый процент объема оперативной памяти V_j.
- Для каждой работы $j \in \mathcal{J}$ определено множество возможных режимов выполнения $\mathcal{M}_j = \{(size_{ji}, \tilde{b_{ji}}, \tilde{p_{ji}}) \mid i \in I_j\}$, где I_j — множество индексов режимов работы j; $size_{ji}$ — количество ядер, необходимых для выполнения работы j в режиме i; $\tilde{b_{ji}}$ и $\tilde{p_{ji}}$ – процент потребления шины данных и количество единиц времени, необходимые ей для полного выполнения в идеальных условиях в режиме i, соответственно.

Постановка задачи

- Имеется множество распараллеливаемых работ $\mathcal{J} = \{1, ..., n\}$, m ядер процессора и два ресурса возобновимого типа – шина данных и оперативная память.
- Для каждой работы $j \in \mathcal{J}$ задан требуемый процент объема оперативной памяти V_j.
- Для каждой работы $j \in \mathcal{J}$ определено множество возможных режимов выполнения $\mathcal{M}_j = \{(size_{ji}, \tilde{b_{ji}}, \tilde{p_{ji}}) \mid i \in I_j\}$, где I_j — множество индексов режимов работы j; $size_{ji}$ — количество ядер, необходимых для выполнения работы j в режиме i; $\tilde{b_{ji}}$ и $\tilde{p_{ji}}$ – процент потребления шины данных и количество единиц времени, необходимые ей для полного выполнения в идеальных условиях в режиме i, соответственно.
- Если в конфигурации c работе j в выбранном режиме i выделен процент шины данных $b_{jc} < \tilde{b_{ji}}$, то она пропорционально замедляется.

Постановка задачи

- Имеется множество распараллеливаемых работ $\mathcal{J} = \{1, ..., n\}$, m ядер процессора и два ресурса возобновимого типа – шина данных и оперативная память.
- Для каждой работы $j \in \mathcal{J}$ задан требуемый процент объема оперативной памяти V_j.
- Для каждой работы $j \in \mathcal{J}$ определено множество возможных режимов выполнения $\mathcal{M}_j = \{(size_{ji}, \tilde{b_{ji}}, \tilde{p_{ji}}) \mid i \in I_j\}$, где I_j — множество индексов режимов работы j; $size_{ji}$ — количество ядер, необходимых для выполнения работы j в режиме i; $\tilde{b_{ji}}$ и $\tilde{p_{ji}}$ – процент потребления шины данных и количество единиц времени, необходимые ей для полного выполнения в идеальных условиях в режиме i, соответственно.
- Если в конфигурации c работе j в выбранном режиме i выделен процент шины данных $b_{jc} < \tilde{b_{ji}}$, то она пропорционально замедляется.
- Необходимо составить такое расписание выполнения распараллеливаемых работ на ядрах процессора, что общее время завершения работ $C_{\max}$ минимально, при ограничении на объем оперативной памяти и заданном частичном порядке на множестве работ.

Алгоритм 1 Правило расчета замедления рабоы $j \in J$ в конфигурации $c \in C$

Шаг 1: Положим процент свободной шины данных $freePercent := 100\%$, текущее количество свободных ядер $freeCoresCount := m$ и текущее количество работ $jobsCount$ равно количеству работ в конфигурации c.

Шаг 2: Пока $jobsCount$ не равно 0, выполнять:

2.1: Вычислить процент шины данных, который может быть выделен каждому ядру: $percent := freePercent / freeCoresCount$.

2.2: Если в конфигурации c присутствует такая работа j в режиме $i \in I_j$, для которой $\tilde{b_{ji}} \leq percent \cdot size_{ji}$, положить:

$b_{jc} := \tilde{b_{ji}}$,

$freePercent := freePercent - \tilde{b_{ji}}$,

$jobsCount := jobsCount - 1$.

$freeCoresCount := freeCoresCount - size_{ji}$.

2.3: Иначе положить $b_{jc} := percent \cdot size_{ji}$ для каждой оставшейся работы j и $jobsCount := 0$.

Шаг 3: Скорость выполнения каждой работы j в конфигурации c пропорциональна отношению $b_{jc}/\tilde{b_{ji}}$.

Алгоритм 2 Жадный алгоритм A_{gr} (идея из [Eremeev & Sakhno, 2020])

Пока присутствуют незавершенные работы, повторять:
1: Выбрать допустмимую конфигурацию работ для выполнения.
2: Вычислить скорости выполнения работ в выбранной конфигурации.
3: Определить, какая из работ завершится первой.

Списочные алгоритмы:

- Алгоритм списочного типа с упорядоченными по невозрастанию длительностей $\tilde{p_j}$ работами.
- Алгоритм списочного типа с упорядоченными по невозрастанию объемов V_j работами.
- Алгоритм списочного типа с упорядоченными по невозрастанию $\frac{V_j}{\tilde{p_j}}$ работами.

Эволюционная стратегия (1+1)-ES

Алгоритм 3 (1+1)-ES

Сгенерировать начальную перестановку π случайным образом.

Пока не выполнено условие остановки, повторять:
1: Построить $\pi' = Mut(\pi)$.
2: Если π' по целевой функции лучше π, то $\pi = \pi'$.

Вернуть π.

В качестве оператора мутации использовались операторы сдвига (shift) и обмена (exchange).

Генерация примеров

Параметры:

- Количество работ (50)
- Количество ядер процессора (8)
- Вероятность генерации маленькой, средней, большой работы

Значения характеристик работ:

- Объем памяти, %: 0-10, 10-20, 20-35
- Количество ядер: 1, 2, 4
- Потребление шины данных в идеальных условиях, %: 0-30, 30-80, 80-100
- Длительность в идеальных условиях: 0-10, 10-25, 25-50

Результаты эксперимента

$$LB = \max_{j \in J} \left(\tilde{p_j}, \frac{\sum_{k \in J} \tilde{p_k}}{m} \right)$$

	A_{gr}	$A_{\tilde{p_j}}$	A_{V_j}	$A_{V_j \tilde{p_j}}$	(1+1)-ES_{sh}	(1+1)-ES_{ex}
min	83.73	89.14	79.46	83.55	**52.92**	**55.84**
avg	244.49	270.33	253.11	226.79	**185.67**	**180.45**
max	529.39	504.48	558.40	504.20	**428.42**	**440.68**

Таблица: Относительные отклонения результатов, найденных разными алгоритмами, от нижней границы (LB), %

(1+1)-ES: 1000 итераций, начальное решением получено алгоритмом $A_{V_j \tilde{p_j}}$

Дальнейшие планы

- Записать математическую модель
- Исследовать вычислительную сложность
- Улучшить нижнюю границу
- Попробовать другие стратегии в жадном и списочных алгоритмах
- Адаптировать генетический алгоритм к новой задаче
- Учесть NUMA-архитектуру в постановке

Спасибо за внимание!