Quicksort Sort: Parallel Analysis

Toby Isaac

CSE 6230, Fall 2017

1 Pseudocode

Note that this is for quicksort as written when project 2 was assigned. Improvements are possible.

Algorithm 1: `sortQuicksort`$(r,\ P,\ n,\ A)$

Data: r is the process's rank;
P is the number of processes (power of 2 in this pseudocode);
n is the number of local keys (may differ between processes);
A is the array of keys.

/* local sort with no special condition, $O(n \log_2 n)$ */
`sortLocal`$(n,\ A,\ d)$
if $P > 1$ **then**
    $p \leftarrow$ `choosePivot`$(r,\ P,\ n,\ A)$
    /* Binary search for where p fits between keys in A ($O \log_2 n$) */
    $n_{\mathrm{lo}}, n_{\mathrm{hi}} \leftarrow$ `split`$(n,\ A,\ p)$
    $n_{\mathrm{new}}, A_{\mathrm{new}} \leftarrow$ `swapQuicksort`$(r,\ P,\ n_{\mathrm{lo}},\ n_{\mathrm{hi}},\ A,\ d)$
    `sortQuicksort`$(r \mod P/2,\ P/2,\ n_{\mathrm{new}},\ A_{\mathrm{new}})$
    /* $A_{\mathtt{new}}$ is sorted, but may have wrong size */
    /* repartition to restore original keys per process */
    `repartition`$(r,\ P,\ n_{\mathrm{new}},\ A_{\mathrm{new}},\ n,\ A_{\mathrm{new}})$

Algorithm 2: `choosePivot`$(r,\ P,\ n,\ A)$

Data: r is the process's rank;
P is the number of processes (power of 2);
n is the number of local keys (same on all processes);
A is the array of keys.

if $r = 0$ **then** $p \leftarrow A[n/2]$;
/* Broadcast is $O(\log_2 P)$ */
$p \leftarrow$ `Bcast`$(p,\ 0,\ P)$
return p

Algorithm 3: `swapQuicksort`(r, P, n, A, d)

Data: r is the process's rank;
P is the number of processes (power of 2);
n_{lo} is the number of local keys less than or equal to the pivot;
n_{hi} is the number of local keys greater than to the pivot;
A is the array of keys.

```
/* Communicating partner in the other half */
```
$q \leftarrow r + P/2 \mod P$
if $r < q$ **then**
  $n_{\text{send}}, A_{\text{send}} \leftarrow n_{\text{hi}}, A_{\text{hi}}$
  $n_{\text{keep}}, A_{\text{keep}} \leftarrow n_{\text{lo}}, A_{\text{lo}}$
else
  $n_{\text{send}}, A_{\text{send}} \leftarrow n_{\text{lo}}, A_{\text{lo}}$
  $n_{\text{keep}}, A_{\text{keep}} \leftarrow n_{\text{hi}}, A_{\text{hi}}$
$R \leftarrow$ `Isend`(A_{send}, n_{send}, q)
```
/* New size unknown, can either get upper bound on possible buffer */
/* or probe for message size, allocate buffer, and receive */
```
$A_{\text{new}}, n_{\text{new}} \leftarrow$ `Probe/Recv`(q)
`Wait`(R)
return $n_{keep} + n_{recv}$, $[A_{keep}, A_{recv}]$

2 Runtime analysis

We want an expression for $T_q(n, p)$, the runtime of parallel quicksort in terms of n, the number of local keys, and P, the number of processes.

First the base case:

$$T_q(n, 1) = T_l(n) := T_{\texttt{localSort}}(n) \in O(n \log_2 n).$$

Then the recursive case:

$$\begin{aligned} T_q(n, P) = & T_l(n) + \\ & (T_{cp}(n, P) := T_{\texttt{choosePivot}}(n, P)) + \\ & (T_{spl}(n) := T_{\texttt{split}}(n)) + \\ & (T_{sq}(n, n_{\texttt{new}}, P) := T_{\texttt{swapQuicksort}}(n, n_{\texttt{new}}, P)) + \\ & T_q(n_{\texttt{new}}, P/2) + \\ & (T_{rp}(n, n_{\texttt{new}}, P) := T_{\texttt{repartition}}(n, n_{\texttt{new}}, P)) \end{aligned}$$

We are going to make some simplifying assumptions:

1. $T_l(n) = C_l n \log_2 n$
2. Since we assume in the code that $T_{cp}(n, P) \in O(\log_2 P)$, we will assume that it is independent of n, and $T_{cp}(n, P) = C_{cp} \log_2 P$.

3. $T_{spl}(n) = C_{spl} \log_2 N$.

4. $T_{sq}(n, n_{\text{new}}, P) = C^0_{sq} + C^1_{sq} \max\{n, n_{\text{new}}\}$, where we include the constant term because we think that latency may be important.

5. We will ignore T_{rp} for the current analysis and come back to it later.

6. We will assume that, based on the quality of the pivot, the largest value of n_{new} over all processes grows by some factor $1 \le \theta < 2$ at each iteration, so we will use $n_{\text{new}} = \theta n$. For example, I think that if the starting keys are uniformly random, and the pivot is chosen uniformly randomly over all keys, we would have $\theta = 3/2$.

Then

$$T_q(n, P) = C_l n \log_2 n + C_{cp} \log_2 P + C_{spl} \log_2 n + C^0_{sq} + C^1_{sq} \theta n + T_q(\theta n, P/2).$$

Expanding this recurrence:

$$\begin{aligned}
T_q(n, P) = {} & C_l \quad n \log_2 \quad n + C_{cp} \log_2 P \quad + C_{spl} \log_2 \quad n + C^0_{sq} + C^1_{sq} \quad n + \\
& C_l \theta \; n \log_2(\theta \; n) + C_{cp} \log_2 P/2 + C_{spl} \log_2(\theta \; n) + C^0_{sq} + C^1_{sq} \theta \; n + \\
& C_l \theta^2 n \log_2(\theta^2 n) + C_{cp} \log_2 P/4 + C_{spl} \log_2(\theta^2 n) + C^0_{sq} + C^1_{sq} \theta^2 n + \\
& \vdots \\
& C_l \theta^{\log_2 P - 1} n \log_2(\theta^{\log_2 P - 1} n) + C_{cp} \log_2 2 + C_{spl} \log_2(\theta^{\log_2 P - 1} n) + C^0_{sq} + C^1_{sq} \theta^{\log_2 P - 1} n + \\
& C_l \theta^{\log_2 P} n \log_2(\theta^{\log_2 P} n).
\end{aligned}$$

Let us introduce to terms in θ and $\log_2 P$,

$$\begin{aligned}
S_1(\theta, \log_2 P) &:= \sum_{i=0}^{\log_2 P} \theta^i = \frac{\theta^{\log_2 P + 1} - 1}{\theta - 1}, \\
S_2(\theta, \log_2 P) &:= \log_2 \theta \sum_{i=0}^{\log_2 P} i \theta^i.
\end{aligned}$$

Then we can write the expansion of $T_q(n, P)$ as

$$\begin{aligned}
T_q(n, P) = {} & C_l (S_1(\theta, P) n \log_2 n + S_2(\theta, P) n) + C_{cp} \tfrac{1}{2} (\log_2^2 P + \log_2 P) + \\
& C_{spl} (\log_2 n \log_2 P + \log_2 \theta \tfrac{1}{2} (\log_2 P - 1) \log_2 P) + \\
& C^0_{sq} \log_2 P + C^1_{sq} S_1(\theta, P/2) n.
\end{aligned}$$

In the case of perfect pivot selection, $\theta = 1$, this becomes a lower bound: (Note that $S_1(1, P) = \log_2 P + 1$ and $S_2(1, P) = 0$.)

$$\begin{aligned}
T_q(n, P) \ge {} & C_l (\log_2 P + 1) n \log_2 n + C_{cp} \tfrac{1}{2} (\log_2^2 P + \log_2 P) + \\
& C_{spl} (\log_2 n \log_2 P) + \\
& C^0_{sq} \log_2 P + C^1_{sq} n \log_2 P.
\end{aligned}$$

In the worst case of always choosing an endpoint pivot, $\theta = 2$, this becomes an upper bound: (Note that $S_1(2, P) = 2P - 1$ and $S_2(2, P) = 2P(\log_2 P - 1) + 1$.)

$$\begin{aligned} T_q(n, P) \leq\ & C_l((2P-1)n\log_2 n + (2P(\log_2 P - 1) + 1)n) + C_{cp}\tfrac{1}{2}(\log_2^2 P + \log_2 P) + \\ & C_{spl}(\log_2 n \log_2 P + \tfrac{1}{2}(\log_2 P - 1)\log_2 P) + \\ & C_{sq}^0 \log_2 P + C_{sq}^1 (P-1)n. \end{aligned}$$