An example of aperiodic task analysis

UBC | EECE 494 | Winter 2007

Sathish Gopalakrishnan

March 6, 2007

In this article we will discuss how one can bound the synthetic or instantaneous utilization and use the stage-delay theorem to decide on the schedulability of aperiodic task systems.

The **stage-delay theorem** states that when the synthetic utilization of a processor has an upper bound of U then the response time of a task T_i with deadline D_i is at most

$$R_i = U\frac{1-\frac{U}{2}}{1-U}D_i.$$

The execution time of the task, using the standard notation, is e_i. *Note that the stage-delay theorem allows us to find an upper-bound on the response time of a task for each stage of a multi-stage (distributed) system. This approach does not require deadline distribution; synchronization between stages is also not required because we can deal with aperiodic arrivals.*

The synthetic utilization of a system is time-varying and at any time instant t, the synthetic utilization $U(t)$ is given

$$U(t) = \sum_{T_i \text{ is current}} \frac{e_i}{D_i}.$$

A task T_i is said to be *current* at time t if it has been released before time t but its absolute deadline has not been reached. To make this statement more precise, let us assume that T_i was released at time r_i. T_i is current at time t if $r_i + D_i \leq t$.

In this analysis for aperiodic tasks with deadlines we use only the *deadline monotonic scheduling* policy, which is a fixed-priority policy. DM scheduling is, in fact, the optimal fixed-priority policy for scheduling aperiodic tasks.

U is called an upper-bound on the synthetic utilization of the processor if $U \geq U(t), \forall t$.

In both periodic task analysis and aperiodic task analysis we need some method to bound the maximum workload that arrives at the processor in order to estimate schedulability. In the case of periodic tasks, this is easy because we can find an upperbound on the amount of work that can arrive over an interval of length L. When tasks are aperiodic, there is no way to determine the maximum workload that may arrive in an interval in a manner similar to periodic tasks. It is for this reason that we need to assume that there is an upperbound on the synthetic utilization.

One method to ensure that this is true is *admission control.* With admission control, we ensure that we never admit (or release a task) if the synthetic utilization will exceed the set bound. (As a result, there is some freedom left to the system designer to choose U.)

Apart from fixing the limit on synthetic utilization *a priori* we can estimate the maximum possible synthetic utilization if we have some information about the workload. We shall explore this idea with the help of an example.

Software as a service is an emerging paradigm for many computing applications. A service provider hosts several applications and the computing platform and clients obtain access to the system to submit jobs and obtain results. The level of service demanded by a client from the service provider is usually agreed upon in a *service level agreement*. The SLA will indicate the amount of work a client can submit over an interval (not necessarily periodically) and will also indicate the response time expected from the service provider. The SLA, thus, provides information about execution times and deadlines and the service provider has to guarantee response times. Clients may pay more money for better response times and the service provider needs to determine the infrastructure to invest in so that all SLAs are met.

Suppose a service provider $\mathscr{S}$ has three clients $\mathscr{A}, \mathscr{B}$ and $\mathscr{C}$. The clients have signed up for a similar application, which is executed on a three-stage pipeline. $\mathscr{A}$ and $\mathscr{B}$ have guaranteed that only one request will be generated at a time; new requests will be placed only after an earlier request is complete and its deadline has passed. $\mathscr{C}$ may need to place up to two requests at a time. If $e_{i,j}$ is the execution demanded by client i at stage j then we have the following information:

$$\begin{array}{lll} e_{\mathscr{A},1} = 1 & e_{\mathscr{A},2} = 10 & e_{\mathscr{A},3} = 3 \\ e_{\mathscr{B},1} = 2 & e_{\mathscr{B},2} = 5 & e_{\mathscr{B},3} = 3 \\ e_{\mathscr{C},1} = 7 & e_{\mathscr{C},2} = 2 & e_{\mathscr{C},3} = 6 \end{array}$$

It is also known that the deadline for tasks from $\mathscr{A}$ are 8 times the total execution time; the deadline for tasks from $\mathscr{B}$ are 6 times the total execution time and the deadline for tasks from $\mathscr{C}$ are 4 times the total execution time. Nothing is specified about the arrival times of tasks (making them aperiodic).

To apply the stage delay theorem, we need to get an upperbound on the synthetic utilization at each stage. Let us consider stage 1. The execution time demanded by $\mathscr{A}$ is at most 1 at this stage. The deadline for a task from $\mathscr{A}$ is

$$D_{\mathscr{A}} = (1 + 10 + 3) \times 8 = 112$$

because the deadline is 8 times the total execution time of a task. The contribution to the synthetic utilization by $\mathscr{A}$ at stage 1 is $\frac{e_{\mathscr{A},1}}{D_{\mathscr{A}}} = \frac{1}{112}$. It cannot be any greater because there is at most one task from $\mathscr{A}$ at a time. Similarly, the maximum synthetic utilization contribution due to $\mathscr{B}$ is $\frac{2}{(2+5+3)\times 6} = \frac{2}{60}$. The maximum synthetic utilization contribution due to $\mathscr{C}$ is $2 \times \frac{7}{(7+2+6)\times 4} = 2 \times \frac{7}{60}$. The maximum possible synthetic utilization at stage 1, therefore, is $U_1 = \frac{1}{112} + \frac{1}{30} + \frac{7}{30} \approx 0.276$.

We can go through a similar process and determine the maximum possible synthetic utilization at stages 2 and 3: $U_2 \approx 0.2393$ and $U_3 \approx 0.2768$.

Using the stage-delay theorem, we can obtain the following upperbounds on the response times of tasks from $\mathscr{A}$ at each of the stages:

$$
\begin{aligned}
R_{\mathscr{A},1} &= U_1 \frac{1-\frac{U_1}{2}}{1-U_1} D_{\mathscr{A}} = 0.3287 \times 112 = 36.8144 \\
R_{\mathscr{A},2} &= U_2 \frac{1-\frac{U_2}{2}}{1-U_2} D_{\mathscr{A}} = 0.277 \times 112 = 31.024 \\
R_{\mathscr{A},3} &= U_3 \frac{1-\frac{U_3}{2}}{1-U_3} D_{\mathscr{A}} = 0.3298 \times 112 = 36.9376
\end{aligned}
$$

The maximum *total* response time for tasks from $\mathscr{A} = 36.8144 + 31.024 + 36.9376 = 104.776$, which means that tasks from $\mathscr{A}$ will meet their deadline.

We can even shorten this process by verifying that the following is true:

$$\sum_{j=1}^{m} U_j \frac{1-\frac{U_j}{2}}{1-U_j} \leq 1,$$

where m is the number of stages (in this example $m = 3$). If the above inequality holds, then all tasks are guaranteed to meet their deadlines. For this example

$$\sum_{j=1}^{3} U_j \frac{1-\frac{U_j}{2}}{1-U_j} = 0.3287 + 0.277 + 0.3298 = 0.9355,$$

and all tasks will meet their deadlines.

Some further comments on the synthetic utilization based tests:

- The test is a sufficient test, not a necessary test. Task sets that fail the test *may* be schedulable.
- When we do not know the exact execution times but we do know that the deadline of the end-to-end task is $k \times \sum_{j=1}^{m} e_{i,j}$ we can *upperbound* the synthetic utilization contribution of task T_i at each stage as $\frac{1}{k}$.
- For admission control, we can admit a task if the new task and the tasks that have already been admitted will meet their deadlines (using the analysis). If not, we can reject a task. Sometimes it is better to reject a task rather than admit it and have it miss a deadline; when a task is rejected we try to schedule it on a different processor and therefore rejection is better than a deadline miss.
- The relationship between synthetic utilization and actual utilization is unclear; even when synthetic utilization is low the actual processor utilization may be rather high.
- When tasks are strictly periodic and relative deadlines are equal to periods, the actual processor utilization is an upperbound on the synthetic utilization of the taskset. (Why?)

Aperiodic task analysis is a powerful tool for estimating the schedulability of systems when information is not easily available or when arrivals are random. On the other hand, aperiodic task analysis can be extremely pessimistic when we know about task arrival patterns. It is much better to use periodic analysis when information is available.

References

1. Tarek Abdelzaher, Gautam Thaker, Patrick Lardieri, *A Feasible Region for Meeting Aperiodic End-to-end Deadlines in Resource Pipelines*, IEEE International Conference on Distributed Computing Systems, Tokyo, Japan, March 2004.

2. Tarek Abdelzaher, Vivek Sharma, Chenyang Lu, *A Utilization Bound for Aperiodic Tasks and Priority Driven Scheduling*, IEEE Transactions on Computers, Vol. 53, No. 3, March 2004.

3. Xue Liu and Tarek Abdelzaher, *On Non-utilization Bounds for Arbitrary Fixed-priority Policies*, IEEE RTAS, San Jose, CA, April 2006.