Systematic Methodology for Measuring Completion Time in Systems

Abstract

This paper presents a systematic methodology for measuring and estimating the completion time of processing systems, which can include single-step processes, parallel steps, or complex multi-stage pipelines. The methodology is based on a general equation that models the completion time as a function of latency, cycle time, and bandwidth constraints. By following a structured four-step procedure, practitioners can accurately determine the performance characteristics of a system and predict completion times for any given input size.

1. Introduction

Performance measurement is a critical aspect of system analysis, whether in data processing, network transmission, or manufacturing systems. Many existing approaches rely on ad-hoc testing with different input sizes, often leading to uncertainty and inefficiency. This paper introduces a structured methodology to systematically measure and derive the key performance parameters of a system, enabling accurate prediction of completion time across varying input sizes.

This methodology is applicable to systems with varying complexities, from single-step processes to parallel or pipelined systems. The central idea is to determine the system’s latency, bandwidth, and cycle time using minimal testing and analysis, avoiding the need for exhaustive testing or arbitrary assumptions.

> **Note:** We use the term **latency** in place of **throughput time** to avoid confusion. The term **throughput** typically refers to the reciprocal of the cycle time, which could lead to ambiguity when discussing system performance.

---

2. The Completion Time Equation

The methodology is based on the following general equation for completion time:

$$\text{Completion Time} = \text{Latency} + \max(\text{Cycle Time} \times (\text{size} - \text{bandwidth}), 0)$$

Definitions of Terms:

- **Latency (L):** The initial overhead or startup time required to process the first unit of input, determined using a 1-byte test.
- **Cycle Time ©:** The time taken to process one unit of input once the system is in steady-state operation.

- **Size (S):** The size of the input being processed, measured in appropriate units (e.g., bytes, packets, tasks).
- **Bandwidth (B):** The maximum number of units that the system can process simultaneously or before queueing effects take over.

This equation accounts for both the latency-driven initial phase and the steady-state phase governed by cycle time and bandwidth constraints. If the system has no bandwidth limitation, the equation defaults to a linear relationship with cycle time.

---

3. Systematic Procedure

Step 1: Determine Latency

- Test the system using a minimal input size (1-byte or 1-unit file).
- Measure the time taken for the system to complete processing.
- The measured time is the system's **latency (L)**.

$$L = \text{Completion Time for 1-byte input}$$

Step 2: Determine Bandwidth

- Test the system using various input sizes (small, medium, and large).
- Identify the point where the completion time begins to increase linearly with input size.

- Use the **Bisection Method** to refine and pinpoint the bandwidth value (B) where this transition occurs.

Step 3: Determine Cycle Time

- Select 4 or more test points at and beyond the bandwidth threshold.
- Perform linear regression on the completion time versus input size (You can use programing, or Excel SLOPE() function) .
  - If the time growth is linear, the slope of the regression line is the **cycle time ©**.
  - If the time growth is not linear, the current equation does not apply, indicating either unexpected system behavior or the need for a different model.

$$C = \text{Slope of the linear regression line (if applicable)}$$

- If non-linearity is detected:
  - Halt further steps and investigate the cause.
  - Either fix the issue or derive a non-linear model to fit the system’s behavior.
- To optimize the process, finding the letency, finding the break point, and performing linear regression by one program. Such program can call the system directly or import the resulting data.

Step 4: Validate and Predict Completion Time

- Apply the determined values of latency (L), bandwidth (B), and cycle time (C) in the completion time equation.

$$\text{Completion Time} = L + \max(C \times (S - B), 0)$$

- Optionally, test with larger input sizes to check for unexpected thresholds or sudden non-linear growth. This step helps confirm the robustness of the model.

---

4. Example Application

Consider a data transmission system with the following test results:

- Latency $L = 2\,\text{seconds}$ (from a 1-byte test).
- Bandwidth $B = 5\,\text{units}$ (determined using the Bisection Method).
- Cycle time $C = 0.4\,\text{seconds/unit}$ (from linear regression beyond the bandwidth).

Predict Completion Time for a 20-unit Input

$$\text{Completion Time} = 2 + \max(0.4 \times (20 - 5), 0)$$

$$= 2 + 0.4 \times 15 = 2 + 6 = 8\,\text{seconds}$$

---

5. Discussion

This methodology provides a practical and systematic approach to estimating completion time without the need for exhaustive testing. It covers key performance characteristics while offering flexibility to handle non-linear behaviors when they arise. The method is general enough to apply to different systems, from single-step processes to complex pipelined or parallel architectures.

Handling Non-linear Behavior:

In cases where completion time does not grow linearly beyond the bandwidth threshold, practitioners can either:

- Investigate and address potential system inefficiencies.
- Develop an alternative model to capture the non-linear relationship between input size and completion time.

---

6. Conclusion

This paper presents a structured, theory-based methodology for measuring and predicting system completion time using minimal testing. By systematically determining latency, bandwidth, and cycle time, practitioners can accurately estimate completion times across a range of input sizes and detect potential system bottlenecks or anomalies. This approach reduces uncertainty and provides a powerful tool for performance optimization across various domains, including data processing, networking, and manufacturing.