The Working Set Model for Program Behavior

Peter J. Denning
Massachusetts Institute of Technology,
Cambridge, Massachusetts
About the paper

• Published in 1968, 1st ACM Symposium on Operation Systems Principles (SOSP)
• 2005, SIGOPS Hall of Fame Award

“This paper introduced the working set model, which has became a key concept in understanding of locality of memory references and for implementing virtual memory. Most paging algorithms can trace their roots back to this work”
Background: Virtual memory

• In the 50s ~ 60s
  – System: ATLAS computer
  – Two-level memory hierarchies: main memory + auxiliary storage
  – Demand paging
  – Backbone for multiprogramming
“When it was first observed in the 1960s, thrashing was an unexpected, sudden drop in throughput of a multiprogrammed system ... I explained the phenomenon in 1968 and showed that a working-set memory controller would stabilize the system ...” – Peter D. Denning
Motivation

• Develop a unified approach to tackle the resource allocation problem: process scheduling and memory management
• However, such unified approach is absent, because there is no adequate model for program behavior
  – Program behavior: a program’s system demand, including *processor* and *memory demand*
Working Set Model - Overview

• Intended to model the behavior of programs, specifically the programs’ memory demand

• What is the working set of a program?
  – Programmer’s standpoint: the smallest collection of information must be present in main memory to assure efficient execution of the program
  – System’s standpoint: the set of most recently referenced pages
Working Set Model - Definition

• Working set information of a process at time $t$ is defined as

$$W(t, \tau)$$

collection of memory pages referenced by the process during the time interval $(t - \tau, t)$, $\tau$ is the working set parameter
Working Set Model - Example

sequences of page references

\[ W(t_1, \tau) = \{2, 6, 1, 5, 7\} \]

\[ W(t_2, \tau) = \{7, 5, 1\} \]

\[ W(t_3, \tau) = \{4, 1, 2, 3\} \]
Working Set Model - Properties

• P1. Size of working set
  – $w(t, \tau)$: number of pages in working set $W(t, \tau)$
  – $w(t, 0)$ is 0
  – $w(t, \tau)$ is a monotonically increasing function of $\tau$
Working Set Model - Properties

• P2. Prediction
  – For **small time separation** \( \alpha \), the set \( W(t, \tau) \) is a good predictor for the set \( W(t + \alpha, \tau) \), because references to the same page tend to cluster in time, the probability (below) is small

\[
Pr[W(t + \alpha, \alpha) \cap W(t, \tau) = \emptyset]
\]

  – On the other hand, for **large time separation** \( \alpha \) (say, \( \alpha \gg \tau \)), \( W(t, \tau) \) is a not good predictor for the set \( W(t + \alpha, \tau) \)
Working Set Model - Properties

• P3. Reentry Rate
  – The rate at which memory page reenters a working set
  – Important property for analyzing page traffic
  – Highly related to the value of $\tau$.
    • E.g. as $\tau$ is reduced, size of working set decreases, the probability that needed pages are not in $W(t, \tau)$ increases.
Working Set Model - Properties

• P3. Reentry Rate (continued)
  – Process-time reentry rate \( \lambda(\tau) = 1/m(\tau) \), where \( m(\tau) \) is the mean time between reentries
  – Real-time reentry rate \( \varphi(\tau) = \lambda(\tau) / (1 + \lambda(\tau)T) \), where \( T \) is the traverse time for fetching a page
  – Total return traffic rate \( \beta M \varphi(\tau) \)
    • \( \beta M \) is the average number of pages for all working sets
    • It estimates the number of pages per unit time returning to main memory
Working Set Model - Properties

• P4. \( \tau \) – Sensitivity
  – Measures how sensitive is the reentry rate \( \lambda(\tau) \) to changes in \( \tau \)

\[
\sigma(\tau) = - \frac{d}{d\tau} \lambda(\tau)
\]

if \( \tau \) is decreased by \( d\tau \), \( \lambda(\tau) \) increases by \( d\tau \sigma(\tau) \)
Choice of $\tau$

• If $\tau$ is “too small”, it may result in high traffic of returning pages
• If $\tau$ is “too large”, it may result in wasted main memory
• Recommended $\tau$: a value comparable to the memory traverse time $T$
Detecting $W(t, \tau)$

- A hardware-based approach is described, but not practical
- Need a software-based approach
- The paper proposes sampling the page table entries at intervals of $\sigma$ seconds, where $\sigma = \tau / K$ ($K$ an integer constant)

“The idea of sampling for used pages was not new ….. What was new was that the window was defined in the virtual time of the program ….”
Detecting $W(t, \tau)$

- Page table entries for detecting $W(t, \tau)$
  
  - $M$ is 1 if and only if page is in main memory
  - $1 \rightarrow u_0$ each time a page reference occurs
  - At end of each sampling interval, pattern in use bits shifted right one position
Memory Allocation

- A program will not be run unless there is space in memory for its working set
- Knowledge of working set size $w(t, \tau)$ with demand paging suffices to manage memory well
- Before running a process, insure that there are enough pages of memory to contain its working set
Scheduling Implementation with Working Set

• Required properties
  – Memory management and process scheduling be closely related activities
  – Sampling of page tables, only on changing working sets, and as infrequently as possible
  – Provide measurements of current working set sizes and processor time consumption for each process
Resource Allocation: A Balancing Problem

• Formulate the resource allocation as a problem of balancing processor and memory demands against available resources

• “memory demand” of process $I$

$$m_i = \min\left(\frac{w_i}{M}, 1\right)$$

where $M$ is number of pages of main memory
Resource Allocation: A Balancing Problem

• But how to define the “processor demand”
• Idea: define “processor demand” to be the fraction of a standard interval a process is expected to use before it blocks
  – Standard interval $A$, chosen to reflect the maximum tolerable response time to a user
  – Conditional expectation function $Q(t_i)$, to predict the amount of process time to elapse before the next interaction
Resource Allocation: A Balancing Problem

• “processor demand” of process $i$

\[ p_i = \frac{Q(t_i)}{NA} \]

Where $N$ is the number of processors

• “system demand” is defined as $D_i = (p_i, m_i)$
Resource Allocation: A Balancing Problem

• Then, a system is balanced if

\[ S = \sum D = (\alpha, \beta) \]

S is dynamically maintained, and measures the total demand of current running list

\( \alpha, \beta \) are constants chose to achieve desired fraction of resource to constitute balance

• To keep the computer system in balance, a “balance policy” can be designed to minimize

( \( S - (\alpha, \beta) \) )
Conclusion

• Define the working set model for program behavior
• Working set model provides an convenient way to determine what information is in use by a computation and which is not
• Working set model enables simple determination of memory demands