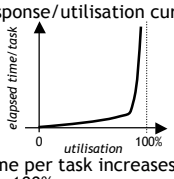
Technology Management and Professional Issues (3001)

System Behaviour

Professor David S. Rosenblum
Department of Computer Science
http://www.cs.ucl.ac.uk/staff/D.Rosenblum/

Response/Utilisation Curves

- Consider a system working with a single task stream
- Typical response/utilisation curve



- Elapsed time per task increases very rapidly as utilisation approaches 100%
- Approximately linear at low utilisation
  - Many short tasks
  - Processes running on a multiprocessor machine

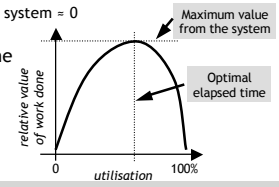
Relative Value/Response Curves

- Relative Value = $\frac{\text{worth of task to the organisation}}{\text{cost of doing it}}$
- Typical relative value/response curve

  - Relative value depends on several factors
    - Efficiency of users
    - Users' waiting time
    - Value of the result as a function of time
      - Example: Predictions
  - Always decreases as a function of time

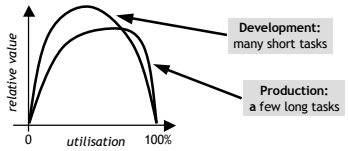
Use of Resources

- Given Utilisation and Relative Value as before
  - Utilisation = 0% means no work done
    - Thus relative value of the system = 0
  - Utilisation = 100% means very long wait for task completion
    - Thus results almost of no value
    - Thus relative value of the system = 0
- In general,
  relative value of work done
  ∝
  utilisation
  x
  relative value/task



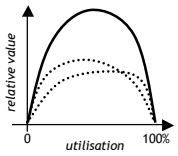
Maximum Value

- Maximum value from system is obtained *at less than* 100% utilisation
- This simple model implies
  - Increased speed leads to increased value
  - But providing a faster system is costlier
- Need to obtain a curve like this for each kind (class) of work carried out



Development versus Production (1)

- Development stream: fast
  - Delay is very expensive
  - Programmers' time, researchers' time
- Production stream
  - Delay is less critical
- Combined curves (by addition)
  - Both streams run at *less than optimal value*
    - One above optimal utilisation
    - One below optimal utilisation



Development versus Production (2)

- Details of combined relative value curve depend on
  - Relative values of the different kinds of work
  - Shape of the value/utilisation curves for them
- Examples
  - Forecasts
    - No value after a critical time
  - Accounting
    - Value drops discontinuously at a critical time
  - Contract
    - Value drops discontinuously if deadlines cannot be met
      - May not drop to 0
      - Modern contracts are often different
- Behaviour is similar if there is a mix of several kinds of work
  - 'All' will be processed *suboptimally*

Summary of Relative Value

- The preceding considerations were all for a *given computer system*
- Recall that relative value = $\frac{\text{worth of task to the organisation}}{\text{cost of doing it}}$
- Different systems will produce different curves
- *The cheaper system is not always the better system*

Cost of Getting a Computer Task Done

- Depends on many factors
  - Cost of computer resources used
    - Equipment, system managers, system administrators
  - Cost of software systems required
    - Compilers, interpreters, applications
  - Cost of software tools required
    - Editors, debuggers
  - Cost of other software
    - Libraries, graphics, data processing, statistics. Modelling
  - Cost of users' time
    - Includes cost of delay if users are blocked
  - Running costs
    - Power, air conditioning
  - Installation and maintenance costs
- The different costs depend on
  - Kind of work, kind of computer system, kinds of users

Typical Kinds of Work

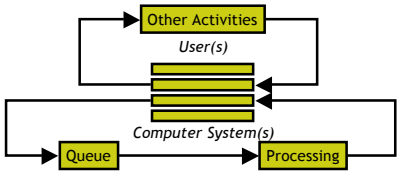
- Computation only
  - Problem is already solved
    - Set parameters
    - Create data input files
    - Run the task
    - Analyse the results
- Programming and debugging
  - Problem is (somewhat) well understood
    - Implement design in code
    - Debug and verify implementation
- Problem exploration
  - Problem not well understood, solution not obvious
    - Develop a solution
    - Check that solution is correct and (reasonably) efficient

Evolution of Computer Work

- 1970s: Change from batch processing on a central computing service to provision of interactive terminals linked to satellite machines coupled to very fast central machines
  - Dominated by programming, debugging, problem development
  - Balance among machine, user and programmer costs
- Today: Many different alternative systems
  - Some very fast (central) machines with a large interactive system
    - Example: 'Big science', engineering
    - Best of the past
    - Users may compete for processor(s), memory, other resources
  - Fast (decentralised) networked workstations and PCs
    - Share files and usage

Interaction Effects

- Model of interaction between user(s) and their system(s)



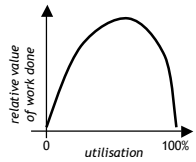
- Two coupled systems: Users, Computer Systems
- To avoid delays, users will often run several computer tasks concurrently

The Concurrency Dilemma

- Because task elapsed (response) times are strongly affected by computer system utilisation, use of concurrency by users can have catastrophic consequences
  - Long elapsed/response times
  - Meaning users each run more tasks concurrently
  - Meaning more tasks in the system
  - Meaning increased utilisation
  - Meaning (greatly) increased response time
  - Meaning the system *grinds to a halt!*
- At *organisational level*, we need to know
  - How users are affected by the computer system
  - How system costs and user costs are affected
  - How computer system behaviour is affected by the users' actions
  - What the resultant system and user costs are

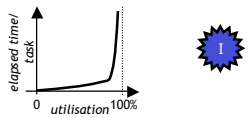
Optimisation

- Unfortunately, computer and user systems are coupled
  - Cannot independently adjust the two systems
  - Behaviour of the two systems feedback on each other
    - Time is wasted by users dependent on the computer system
    - Computer system is sensitive to user behaviour that increases utilisation beyond the optimum



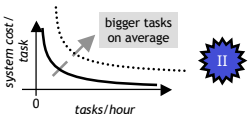
System Costs

- System costs are determined by many of the previously discussed considerations
- For utilisation, we expect:



I

- In addition, if we can run more tasks per hour on the system, the (fixed) cost of the system per task will be reduced



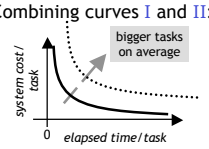
II

Cost per Task as Function of Elapsed Time

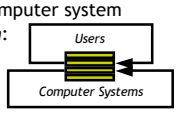
- Naively, we might expect system cost per task to decrease as task elapsed time decreases, since task completes more quickly
- However, this is *not* the case. Consider the extremes:
  - Large number of tasks per hour
    - Means high utilisation
    - So by curve I, this means *long elapsed time*
    - And by curve II, this means *low system cost per task*
  - Small number of tasks per hour
    - Means low utilisation
    - So by curve I, this means *short elapsed time*
    - And by curve II, this means *high system cost per task*
- Computer system costs/task decreased by running more tasks/hour
- Management often wrongly seeks to achieve this, even though this poorly serves the organisation as a whole

Computer System Costs per Task

- Combining curves I and II:

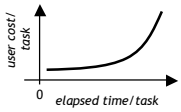


- Meaning we know how costs are affected by changes in the behaviour of the computer system
- But for the *organisation*:

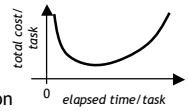


User Costs per Task

- Time wasted waiting for the computer system means number of tasks per day decreases



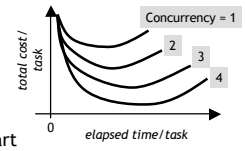
- Meaning for total *organisational* costs, we add the curves and get



- At low computer system utilisation
  - Running several tasks concurrently reduces time wasted for users
  - But eventually, at high utilisation, time will be wasted as elapsed or response times become very long

Effect of Concurrency on Costs

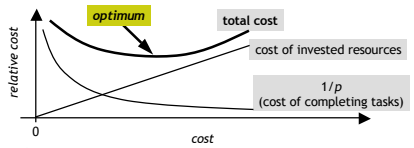
- Concurrency induces a *family of curves* with *diminishing returns* as the number of concurrent tasks increases
- Exacerbated by user stop/start overhead as *user's* concurrency becomes too high
- Management often wrongly seeks to make users work this way, even though *in the long term* this poorly serves the organisation as a whole
- Total cost/task eventually increases
  - Meaning users are unhappy, not working conscientiously, producing poor quality results



Other Factors

- The organisation is concerned with the cost of completing an activity, so mistakes, re-runs, re-work and so on must be taken into account
  - Can increase probability *p* that a task is successful through
    - Debugging aids, libraries, applications
    - Mathematical, numerical, graphics, statistics, visualisation packages
    - Good interface to system/programming environment
    - Good, up-to-date facilities (equipment and software)
    - Staff training
- But all of this costs money

Optimum of Costs



- At the optimum
  - Users have many (but usually not all) of the facilities they want
  - Facilities are well used
    - *But not too well used*
- But it is often difficult to determine *a priori* whether certain facilities are required or not in order for the work to be completed correctly

Allowing for Future Change

- The organisation must also allow for future changes
  - This is *in addition* to getting the task done
- According to M.M. 'Manny' Lehman, we can identify
  - *Progressive part* of the work (PP)
    - The work required to get the task done
    - Deals with increased functionality
  - *Anti-regressive part* of the work (AR)
    - Promotes ability to handle future changes
- Example: Software Development
  - PP = adding functionality, AR = comments and documentation
- Example: Education
  - PP = acquiring skills, AR = knowledge and understanding

UCL ComputerScience