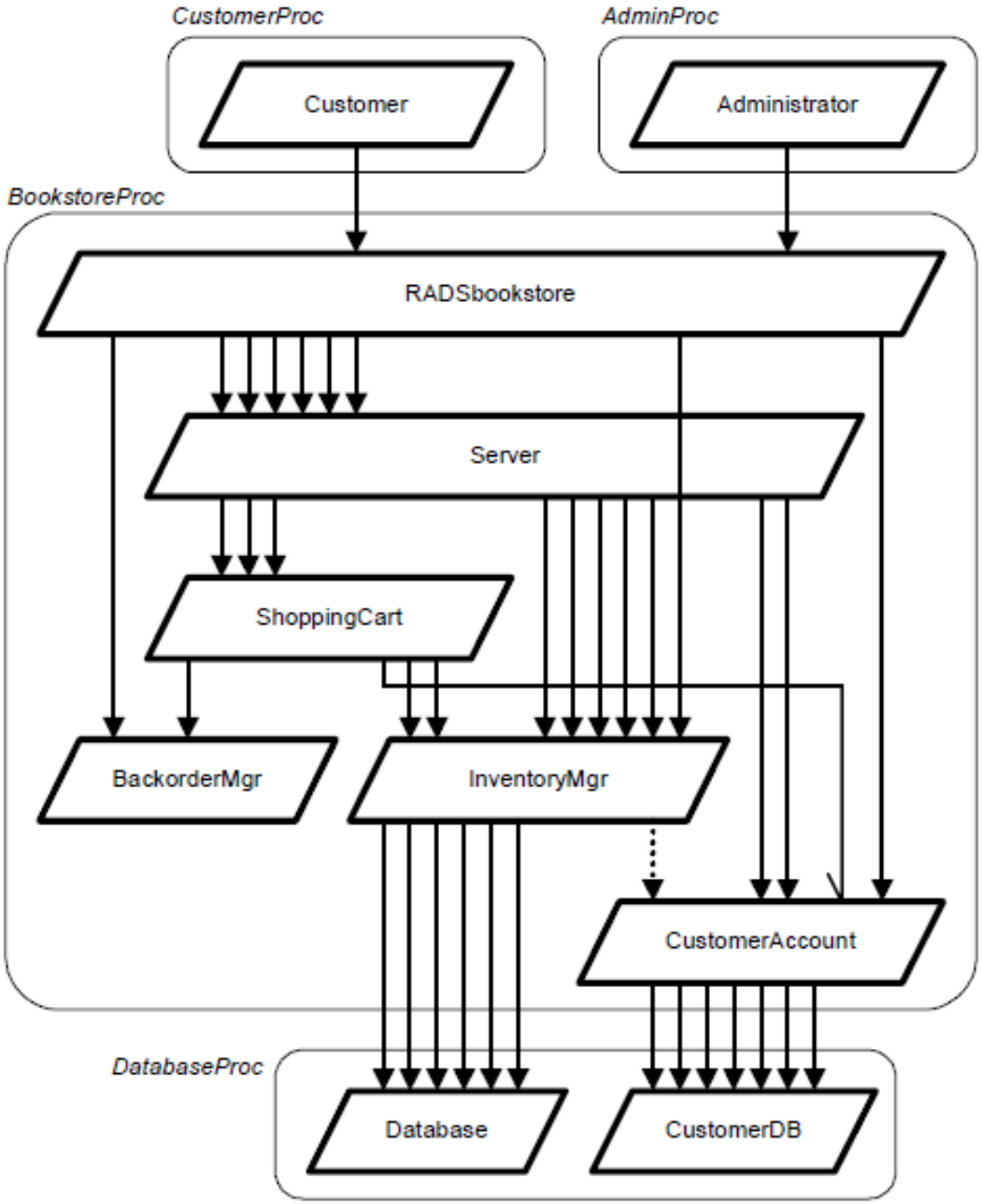
Layered Queueing Network Modeling of Software Systems

Some Examples

1. A Generic Web Service System: Bookstore2000

Murray Woodside
5201 Canal Building

Bookstore2000



A Bookstore

- many Customers, one Administrator
- Bookstore is front end (e.g. web server)
- Server is the Application for Customers
- Browsing, sales and inventory administration all go through the Inventory Manager

The model was generated automatically from nine Use Case Map scenarios

System Components and Operations

- RADSbookstore: presentation (web server)
- Server: application logic for customers (browse, buy)
- ShoppingCart: purchase logic, including inventory update
- BackorderMgr: keeps track of back-orders for out-of-stock items, that have been purchased for later delivery
- InventoryMgr: keeps track of stock, updated by sales and (by the Administrator) for purchases
- Database: database of descriptions of products for sale
- CustomerAccount: server that manages viewing, updating and retrieving customer account data (password, shipping address, credit card)
- CustomerDB: database of customer account data.

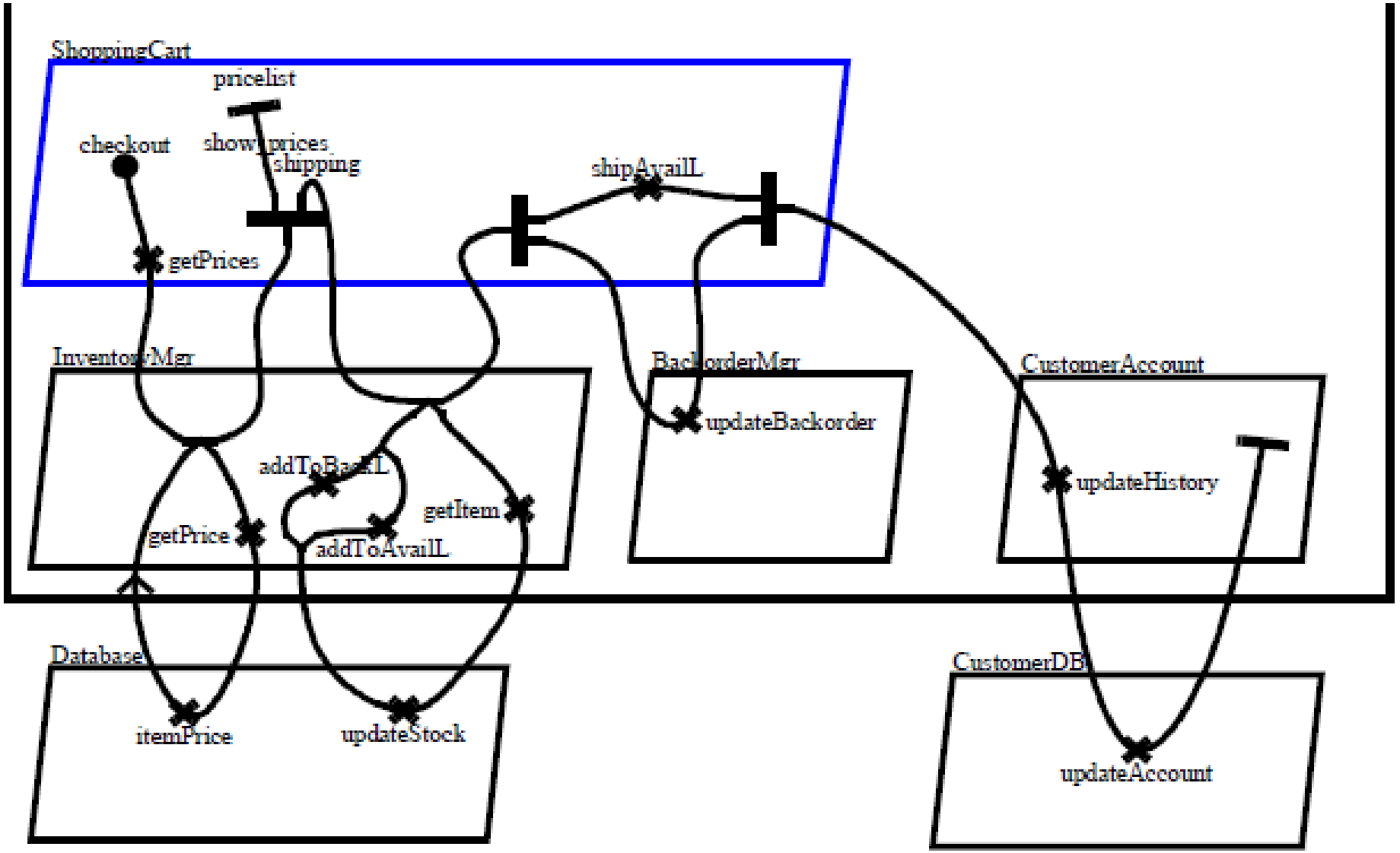
Scenarios: Customers

- *browse*: a list of items is retrieved from the database
- *view_item*: a description is retrieved from the database
- *add_to_cart*
- *remove_from_cart*
- *checkout*
- *login*: the customer's login is verified in and if valid the customer's account information is retrieved from the database
- *register*: a new customer is registered with the bookstore

To support 100 active customers with a think time of 3 sec.

Requirement: a mean response time of one-half sec for all responses

The Checkout Scenario



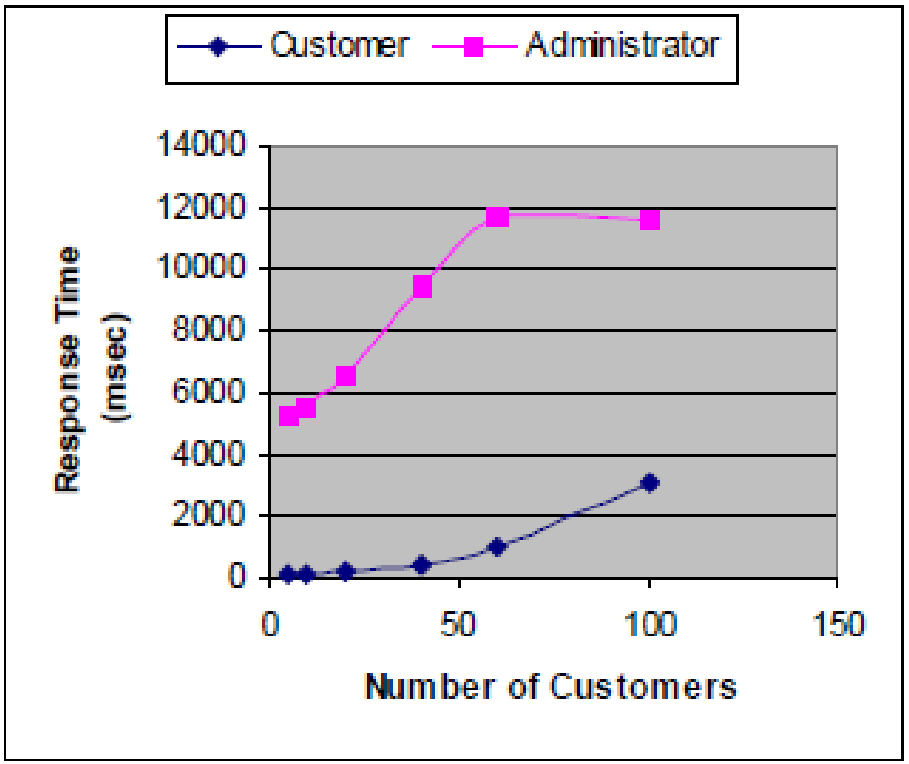
Scenarios: Administrator

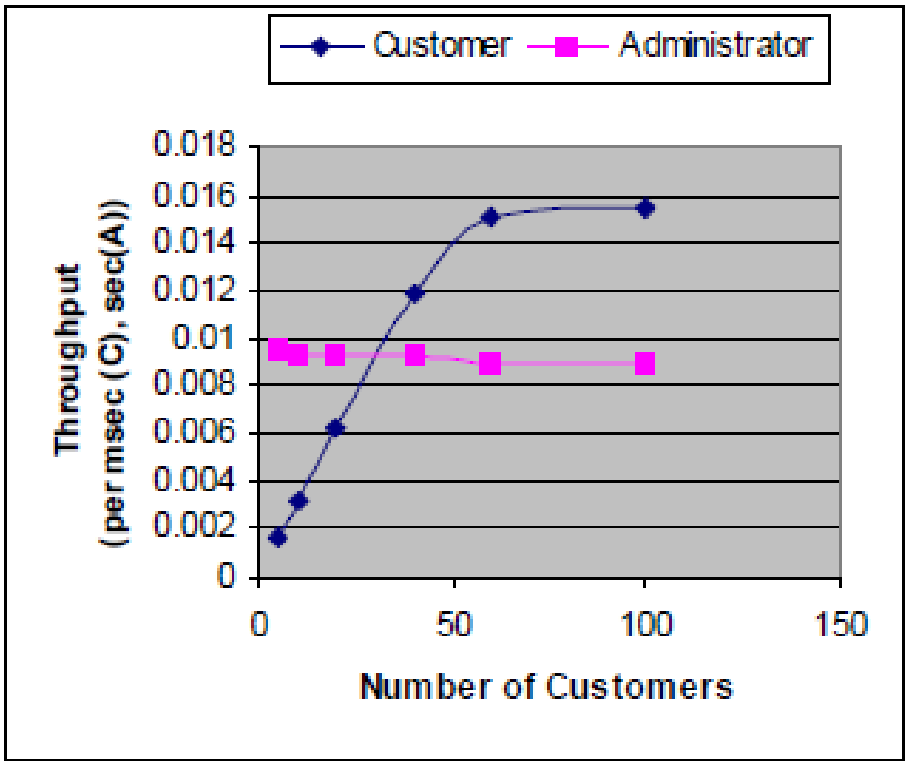
- *update_inventory*: the stock for each item is updated
- *fill_backorders*: outstanding backorders are filled (following an inventory update)

Frequency: modeled as one admin with a think time of 100 sec. We suppose the administrator comes and goes; we model an active period.

Requirement: less strict as the processing may be heavy, but desired to be less than 10 sec on average.

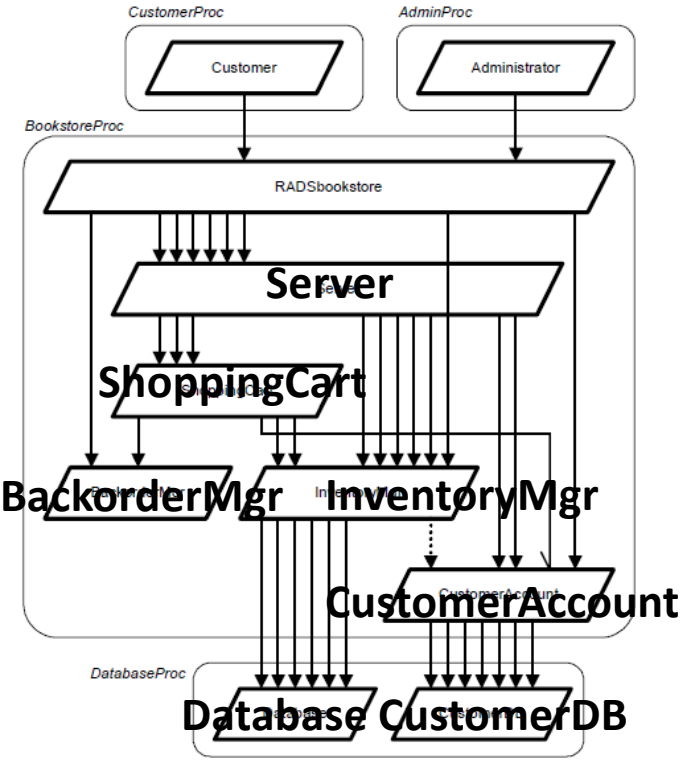
Initial Results





- Response requirements are met at 50 customers, not beyond.

Diagnosis

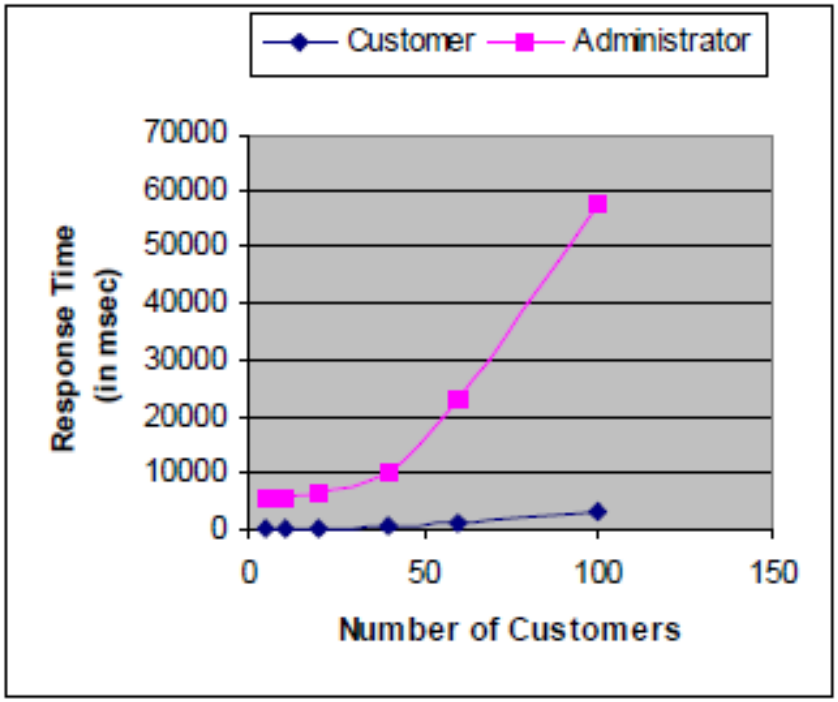


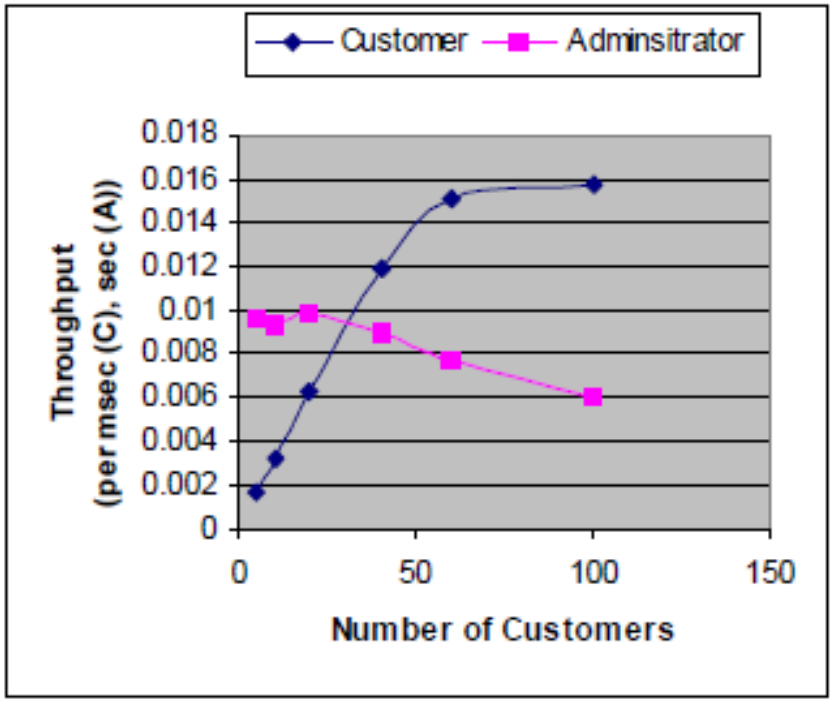
- Customers are queueing at the Server, which is saturated
- InventoryMgr is saturated, and Database is at 80%
- above 50 customers, Admin is not affected by increasing customer traffic... thus, it has a different bottleneck.

- Customers see a bottleneck chain Server-InvMgr-Database, Admin see just InvMgr-Database

More Server Threads

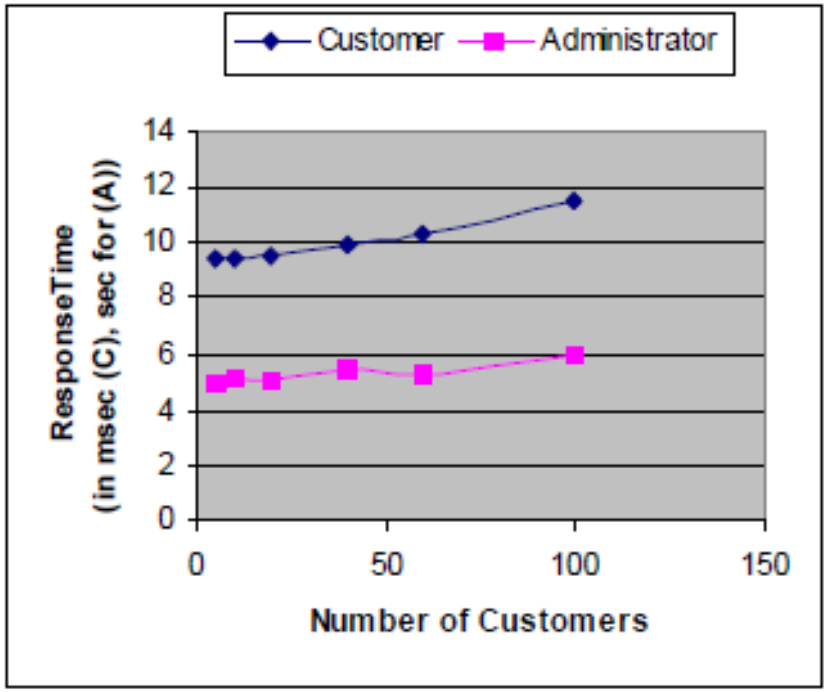
- superficial response to customers queueing at Server
  - go from 5 to 50 threads
- ignores the software bottleneck at InventoryMgr, shadowed by Database
- ineffective: customers not improved, and admin now much worse at high customer loads (flooded by customers)

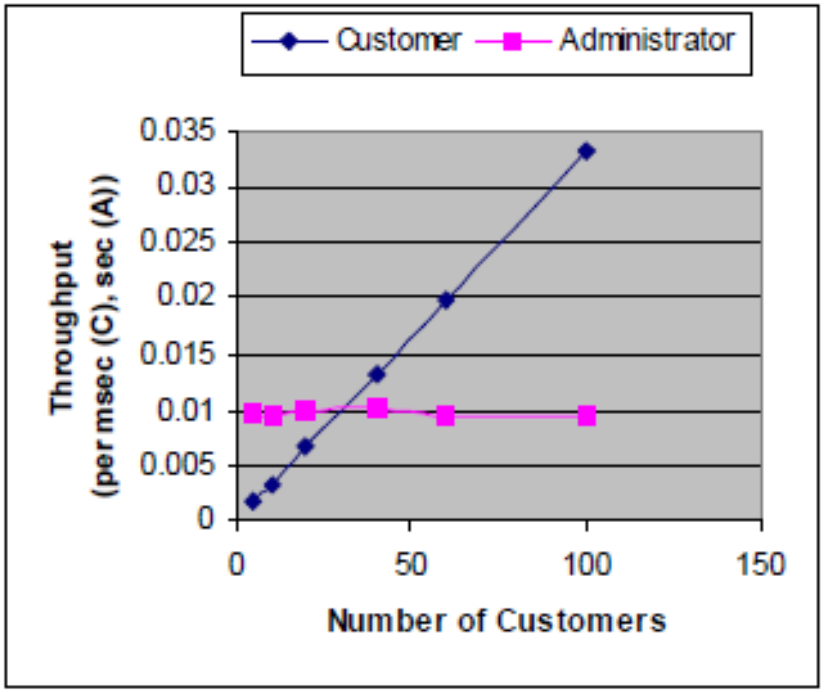




Better: Partition the Database

- Reasoning: adding threads at InventoryMgr will move the bottleneck to Database with only a small improvement
- Database is hard to improve BUT
  - Customer browsing accesses data that changes rarely
  - Put this data in a Catalog database, separate from an Inventory database. MAGIC!





Another Idea: Partition the Server

- The initial configuration has 5 Server threads. If one of these is dedicated to the administrator it solves the administrator's delay problem
  - the thread reduction for customers does not impact their response time, because it is really controlled by the InventoryMgr.
  - the 100% availability of a thread for the administrator (one thread for one user) removes his waiting for the Server
- However this does not improve responsiveness for the customers. More threads does.