Cheaper load testing using fewer simulated users

Pim Kars, Ordina ICT B.V., Software Performance Engineering Group

ORDINA
CONSULTING | ICT | OUTSOURCING

ORDINA.OPGELOST.

Overview

- Short introduction to load testing
- Why reduce the number of simulated users?
- What’s the bottom line?
- How to reduce the number of simulated users?
- Example
- Limitations of the approach
- Conclusion

Load testing

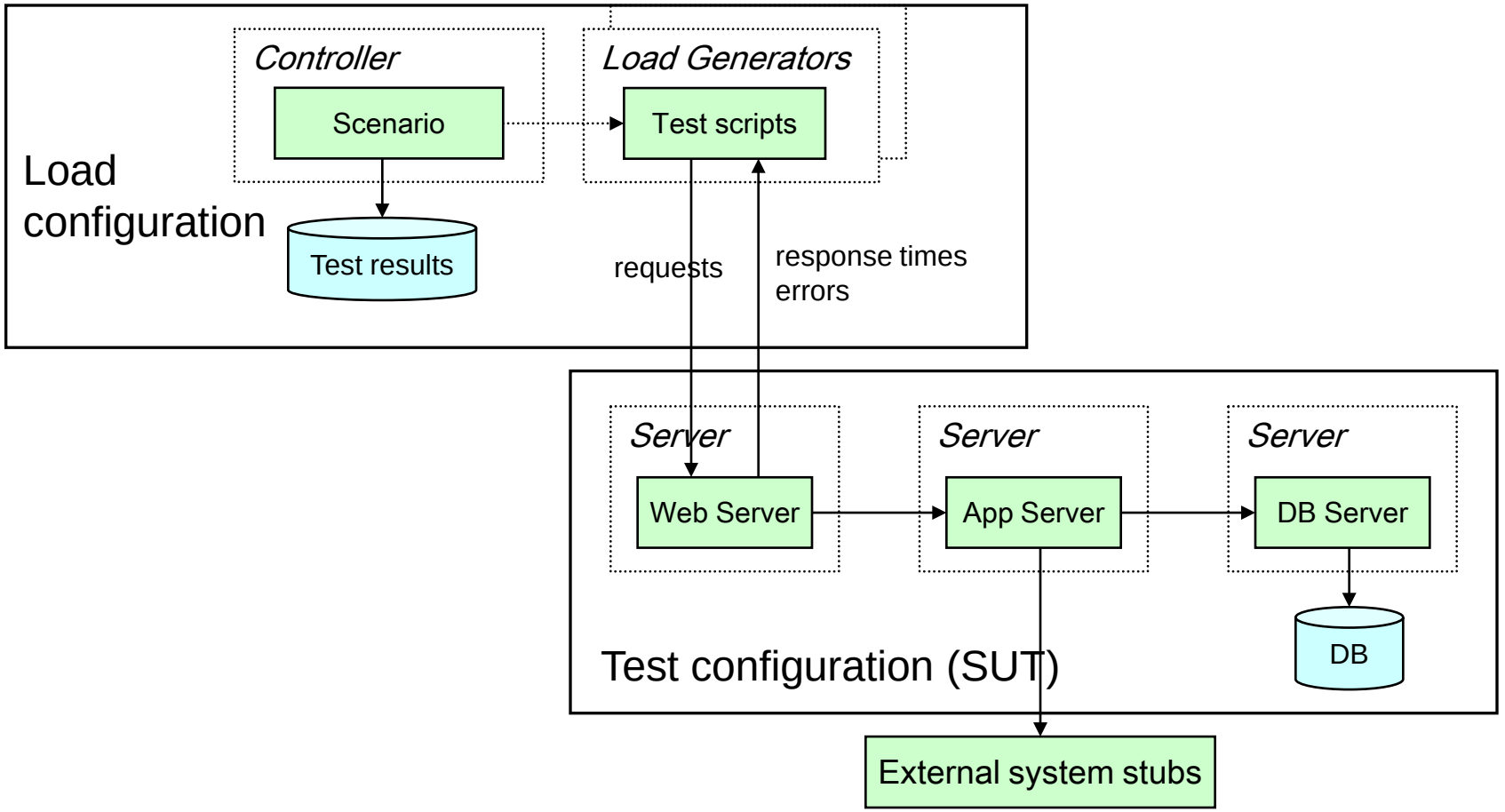
Real life: users exercising functions of an application.

Load testing:

- a load generator tool simulates users by virtual users aka *Vusers* (threads).
- the load generator tool runs on 1 or more servers depending on the number of Vusers.

Example: Web application for order entry

Load test configuration



Load testing terms

A load test run usually has 3 phases:

- *Ramp up period*: slowly build up the load by starting Vusers.
  The duration is linear in the number of Vusers.
- *Stable period*: measurements are taken from this phase.
- *Ramp down period*: gradually stop the Vusers.

Ramp up and ramp down are necessary transition phases,

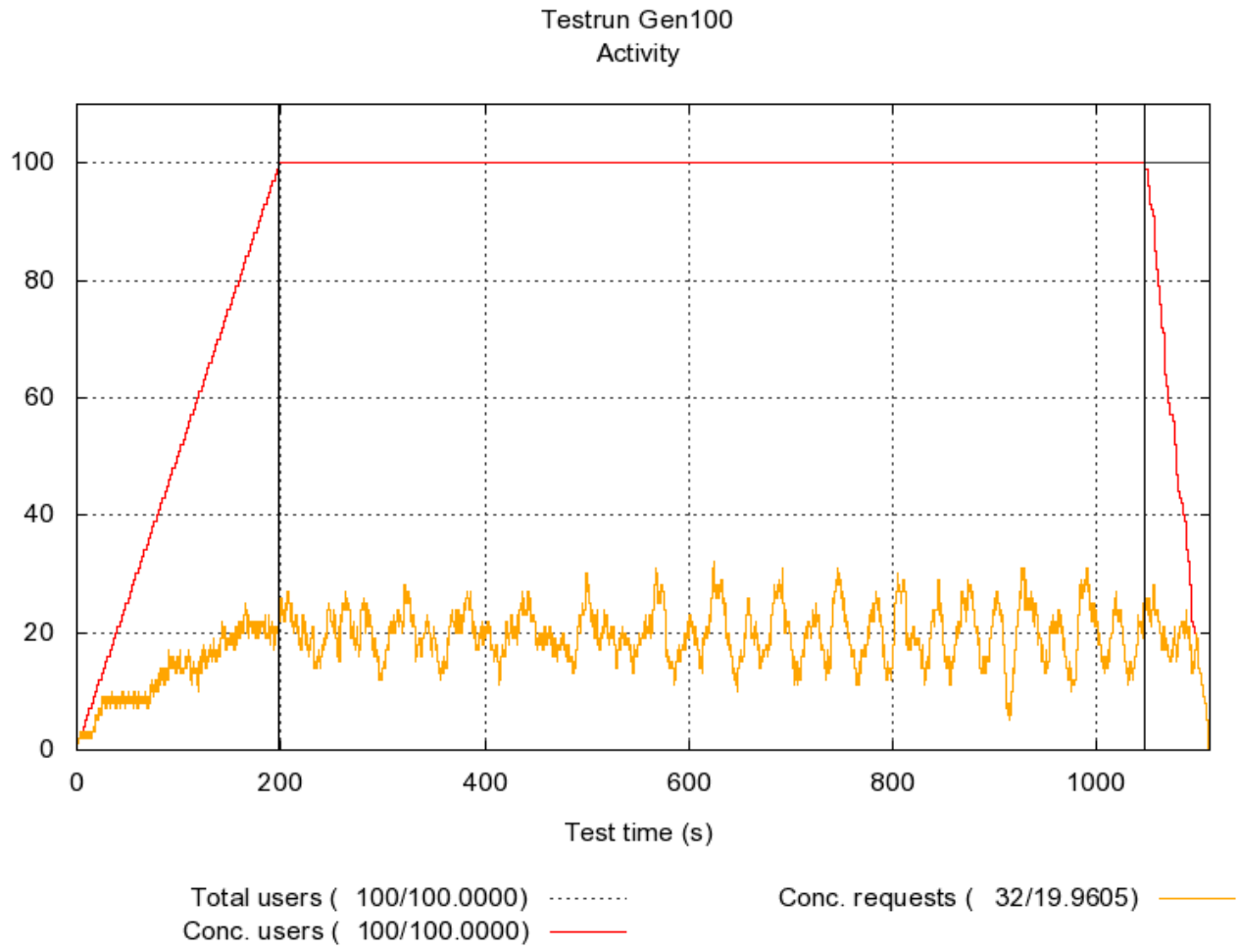
but not useful for measurements.

Example on next slide

Important notion:

The *concurrency level* is the number of concurrent requests in progress.

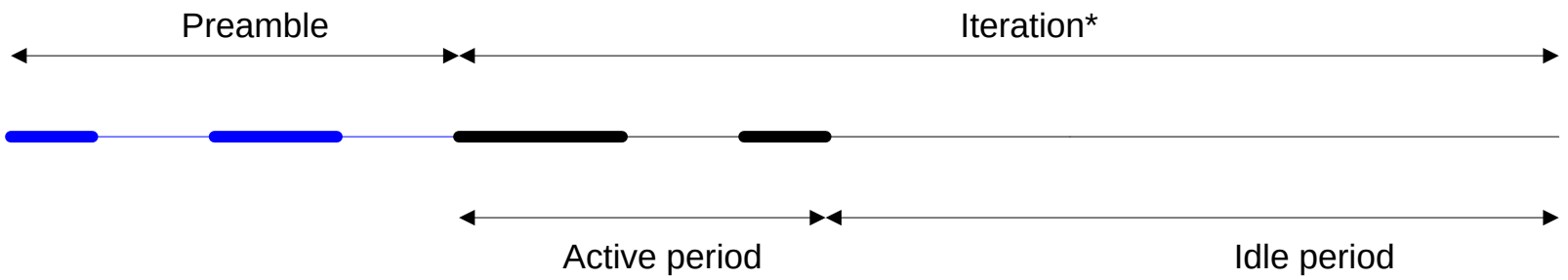
Vusers and concurrency level: 100 Vusers



Assumption about (V)user behaviour

- Typical (V)user behaviour consists of
  a *Preamble* (login etc), a sequence of *Iterations, a Postamble* (logout etc).
- Each *Iteration* consists of
  - an *Active period:* requests with realistic think times in between, and
  - an *Idle period*

Important for this discussion:
useful if the Active period is (much) smaller than the Idle period, and
think times within the Active period are small.



Load level for the basic functionality: Order entry

The (peak) load for Order entry as specified by the customer:

- *Number of users*: 100 users
- *Frequency of use* (per user): 1 order per hour (iteration length = 60 min)

How are users simulated by Vusers?

- Standard practice keeps it simple: use one Vuser for each user.
- The throughput (number of orders handled per hour) is the same for
  100 Vusers handling 1 order /hour
  50 Vusers handling 2 orders/hour
  25 Vusers handling 4 orders/hour
  20 Vusers handling 5 orders/hour
- Suppose it only takes 12 minutes to enter an order (active period = 12 min).
  Then each Vuser is idle 80% of the time!
  It could handle up to 60/12 = 5 orders per hour.

Benefits of reducing the number of Vusers

Question: why reduce the number of Vusers?

Answer: because it is much more efficient to use fewer Vusers than users!

Reducing the number of Vusers saves both time and money, due to

- *Shorter ramp up and ramp down periods* (both are linear in the number of Vusers), thus reducing the length of a test run.
- *Less resource consumption* (e.g. memory, network connections) on the load generators. A test can be run on simpler and often fewer load generators.
- *Less license fees* when a commercial load generation tool is used. The price of a commercial load generator increases with the number of Vusers.

For performance engineers the most important reason is shorter test time.
We often run a lot of test runs to examine the behaviour under varying conditions.
Shorter test time means we can do more test runs in a limited time frame.

For customers less license fees can be very appealing!

Basic idea

As we have seen: the throughput is the same for

100 Vusers handling 1 order /hour
50 Vusers handling 2 orders/hour
30 Vusers handling 3,3333 orders/hour
25 Vusers handling 4 orders/hour
20 Vusers handling 5 orders/hour

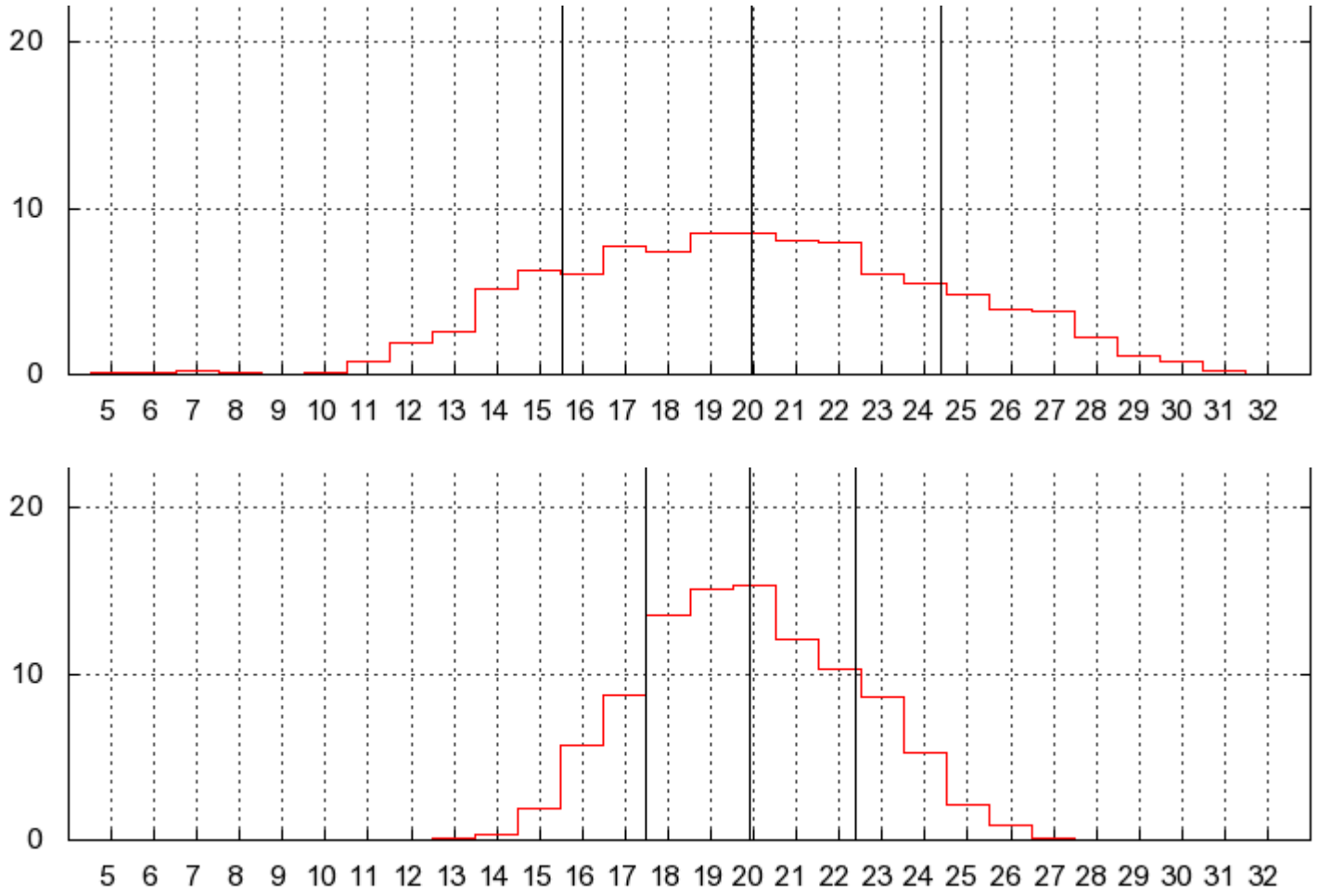
The *average* concurrency level is also the same.

So what's against using only 20 Vusers?

Answer: the *fluctuation* in the concurrency level might be much lower!
Fluctuation in the concurrency level will have an effect on the measured response times.

Example on next slide.

Concurrency level (time%): going from 100 to 50 Vusers



Programme

How to deal with the fluctuation of the concurrency level?

- Determine the concurrency level distribution *C*,
  where *C(k)* is the probability that the concurrency level is *k* ,
  i.e. *k* users are active.
- Determine how to preserve 'enough' fluctuation in the concurrency level
  where fluctuation is interpreted as the standard deviation of *C*.

Notation:

- *N* = number of users
- *A* = length of the active period
- *I* = length of an iteration
- *p* = probability that a user is active at some point in time = *A / I*
- *q* = probability that a user is idle at some point in time = *1 – p*

Concurrency level distribution *C(k)*

Assumption: each (V)user starts randomly within an iteration.

Then the probability *C(k)* that *k* users out of *N* are active at the same time equals:

$$C(k) = \binom{N}{k} \cdot p^{k} \cdot q^{N-k}$$

Conclusion: *C(k)* is a *binomial distribution* with parameters *N* and *p.*

Consequences for the concurrency level:

- Average
  $\mu = N \cdot p$
- Standard deviation
  $\sigma = \sqrt{N \cdot p \cdot q}$

Apologies for the horrible formulae

From users to Vusers

Notation (primes indicate parameters for Vusers):

- N' = number of Vusers
- f = reduction factor = N' / N , with $0 < f \leq 1$ ($f = 1$ means no reduction)

Then

- I' = length of a Vuser iteration = $f \cdot I$
- p' = probability that a Vuser is active = $A / I' = p / f$
- q' = probability that a Vuser is idle = $1 - p'$

Note: if $f < 1$, then $p' > p$ and $q' < q$.

Then

- Average concurrency level doesn't change:
  $\mu' = N' \cdot p' = N \cdot p = \mu$
- Standard deviation does change:
  $\sigma' = \sqrt{N' \cdot p' \cdot q'} = \sqrt{N \cdot p \cdot q'} < \sigma$ if $f < 1$

Conclusion: real reduction means a smaller standard deviation.

Idea

Idea: since we cannot preserve the standard deviation, we put a lower bound on it.

Let β be some prescribed bound with $0 < \beta \leq 1$.

Question: what is the minimal f that preserves at least a fraction β of the standard deviation, i.e. $\beta \leq \sigma' / \sigma$?

Simple algebra shows that f should be at least

$$f_{-} = \frac{p}{1 - \beta^2 \cdot (1 - p)}$$

Slight complication: the resulting value for $N' = N \cdot f$ must be an integer.

So instead of $N' = N \cdot f_{-}$ we take:

$$N' = \lceil N \cdot f_{-} \rceil$$

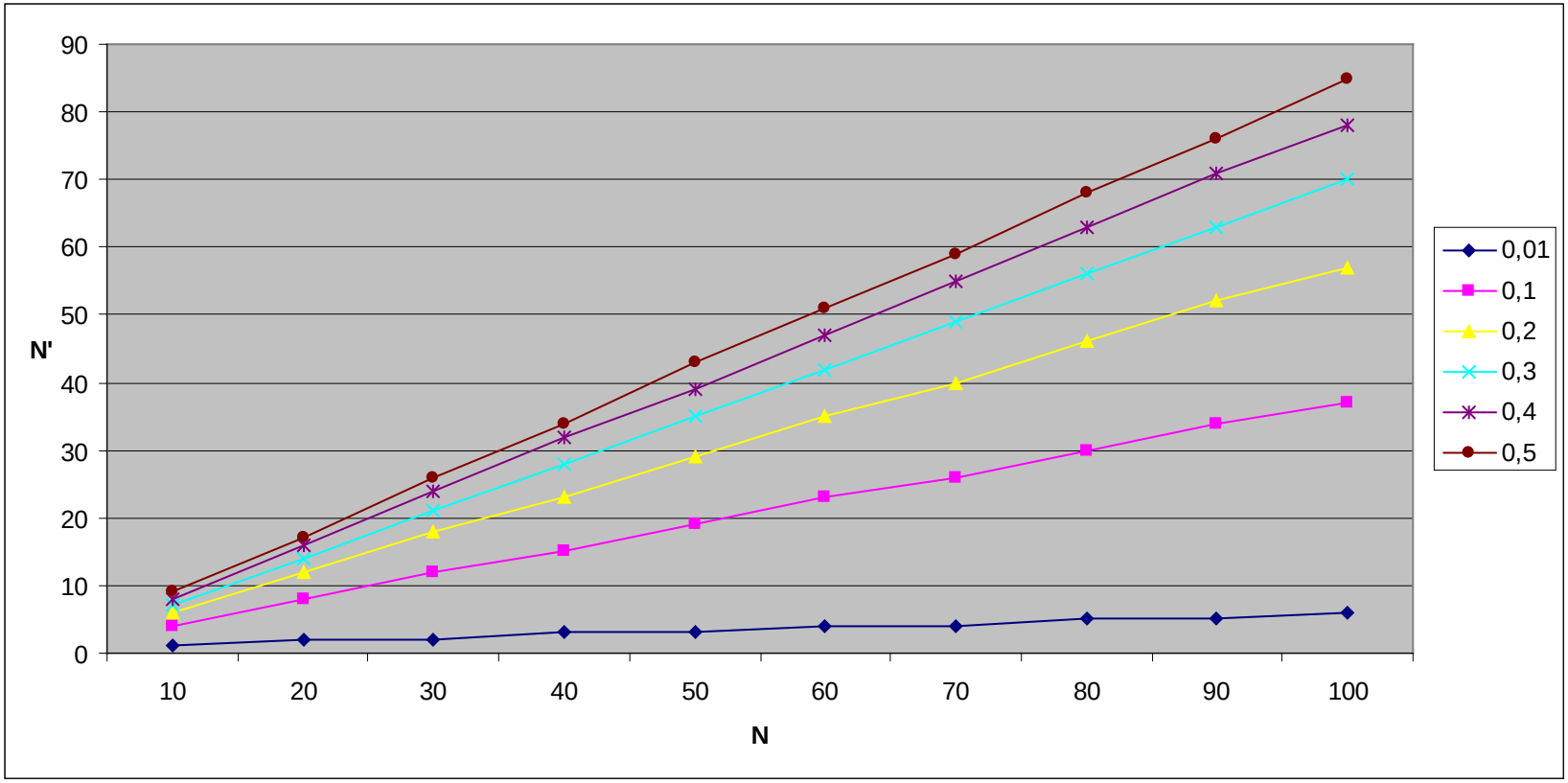
Then f and I' are fixed by previous formulae, and the true reduction in standard deviation is:

$$\frac{\sigma'}{\sigma} = \sqrt{\frac{f - p}{f \cdot (1 - p)}}$$

Result

Example: $N = 100$, $p = 0.2$ gives $N' = 57$ when $\beta = 0.90$.

N' for varying values of N and p when $\beta = 0.90$:



Limitations and caveats

- Sometimes no reduction is possible.
  A trivial example is $N = 1$.
- Extra testing is needed to validate that 100 simultaneous users can be handled by the SUT.
  - Typical problem: insufficient resources, e.g. number of threads or connections.
  - This limitation might not show up in the test with 50 users.
  - Usually one simple test (each user logs in, waits for the other users, and logs out), is sufficient, but it depends on the SUT whether this is sufficient.
- The initial estimate on the length of the active period A is determined with a trial test.
  It depends on the responsiveness of the SUT, so its value should be verified after each test.
- One should verify that the real and simulated concurrency level are binomially distributed.

Conclusion

Shown

- A theory that allows to reduce the number of Vusers, thereby saving time and money.
  It might help to convince customers.
- The theory may yield a considerable reduction, particularly for small p or β.
  In one customer case we reduced 507 users to 51 Vusers, a reduction of 90%!

Extension

- We assumed that (V)users start randomly within each iteration.
  This assumption produces a tractable theory,
  but what happens for other concurrency level distributions?
  In practice, we often try to produce a load with almost no fluctuation at all…

Even though the mathematics is simple, the variables have interesting mathematical properties, if you are into that…

QUESTIONS!?

Ordina
Ringwade 1
3439 LM Nieuwegein
Tel. 030 663 .. ..
www.ordina.nl