Some models of representation of two parallel FIFO-queues and their optimal control

by

Eugene A. Barkovsky

(IAMR KarRC RAS)

RuFiDiM 2014

Introduction

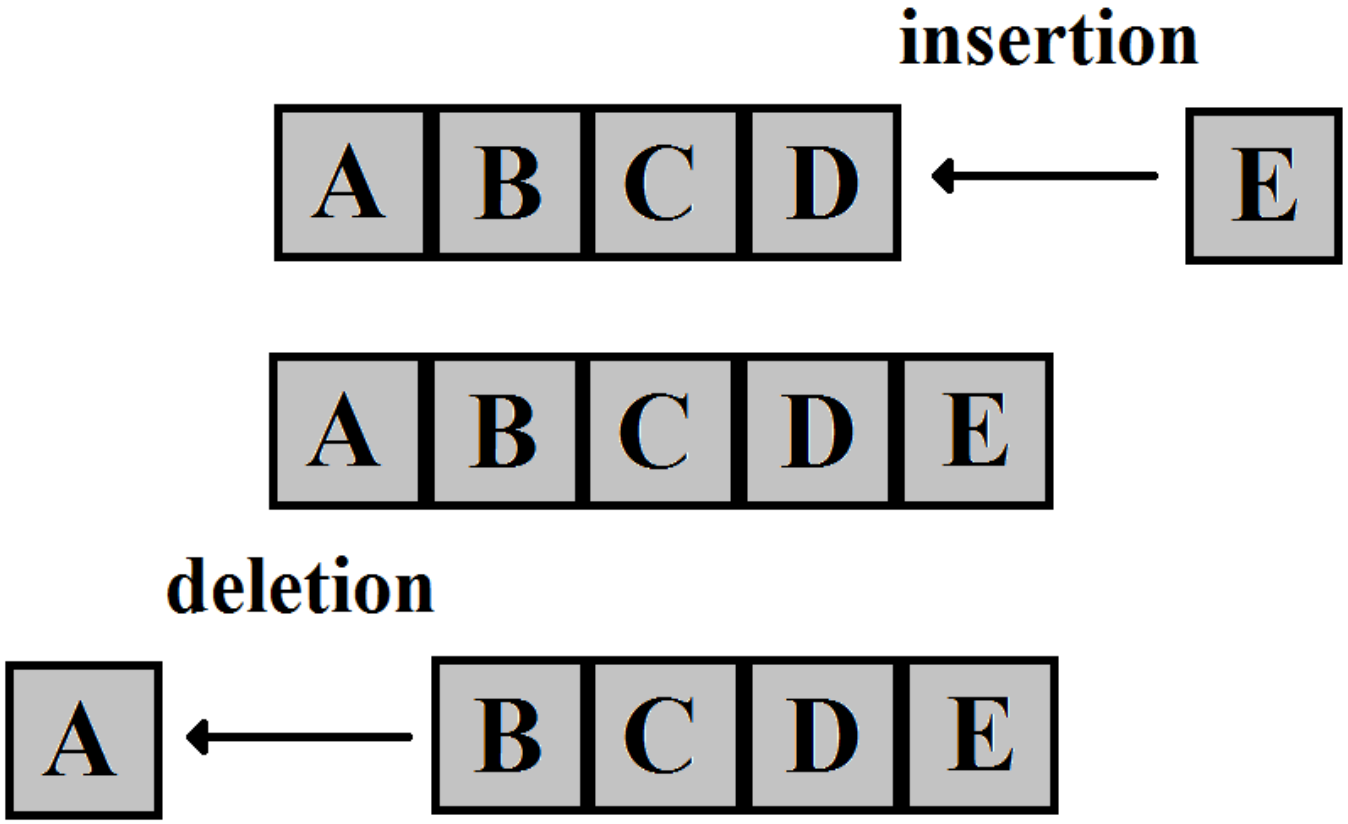
In many applications and tasks, such as the development of various network devices and embedded operating systems, it is required to work with **multiple FIFO-queues** located in **the shared memory** space. Mechanism of paged virtual memory is not used here, and the entire operation occurs in multiple memory pools. The number of queues in such devices can reach several hundreds and thousands.

To represent **FIFO-queues** different software or hardware solutions are used.

Introduction

Here we propose a mathematical models and solve the problems of optimal partitioning of shared memory for **two FIFO-queues** when they are moving **sequentially** and in **a circle one after another** (A.V. Sokolov, 2002) on the assumption that the operations are performed by the principle described by **R. Sedgwick** (1999), but it is possible, along with **the serial**, **the parallel execution** of operations on queues with given probabilities.

FIFO



FIFO (First In – First Out) is a method for organizing and manipulating a data buffer, where the oldest (first) entry, or "head" of the queue, is processed first.

RuFiDiM 2014

R. Sedgwick's principle

Operations with queues are performed by the following scheme:

- **insertion** of an element is performed on **the odd step**;
- and **deletion** — on **the even step**.

Some probabilities of operations performed with queues are known.

Statement of the problem

Suppose that in the memory size of m we are working with **two FIFO-queues** with elements of fixed size in one conventional unit. Then:

- p_1, p_2 and p_{12} — probabilities of **insertion** in queues;
- q_1, q_2 and q_{12} — probabilities of **deletion** from queues;
- r_1 and r_2 — probabilities of operations that **do not change the length** of queues.

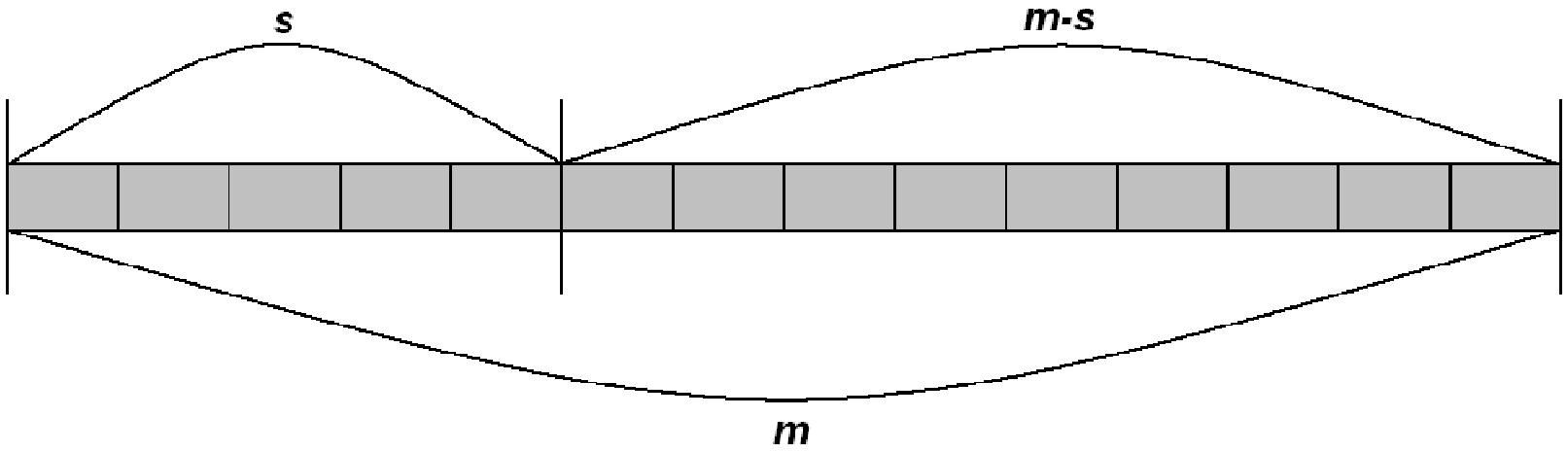
- insertion **in I**, with the deletion **from II** queue — $p_1 q_2$;
- insertion **in II**, with the deletion **from I** queue — $p_2 q_1$;
- insertion **in I** queue — $p_1 r_2 + p_{12} q_2$;
- insertion **in II** queue — $p_2 r_2 + p_{12} q_1$;
- insertion in parallel **in both** queues — $p_{12} r_2$;
- deletion **from I** queue — $q_1 r_1 + p_2 q_{12}$;
- deletion **from II** queue — $q_2 r_1 + p_1 q_{12}$;
- deletion in parallel **from both** queues — $q_{12} r_1$;
- execution of opposite operations, that **preserve** queues status, — $r_1 r_2 + p_1 q_1 + p_2 q_2 + p_{12} q_{12}$.

In a regular Markov chain there is **a limit vector** $\boldsymbol{\alpha}$. This vector is a solution of the equation $\boldsymbol{\alpha * P = \alpha}$, where $\boldsymbol{P}$ is **the transition probability matrix**. According to the law of large numbers for regular Markov chain, element of the vector $\boldsymbol{\alpha_i}$ is the portion of time that the process spends in state $\boldsymbol{i}$.

Therefore, to find out portion of time when queues are **overflowing**, it is necessary to sum elements of the vector $\boldsymbol{\alpha}$, corresponding to those states where newly arriving items for queues are lost.

Our task is to find a vector $\boldsymbol{\alpha}$, where the sum of these states is **minimal**.

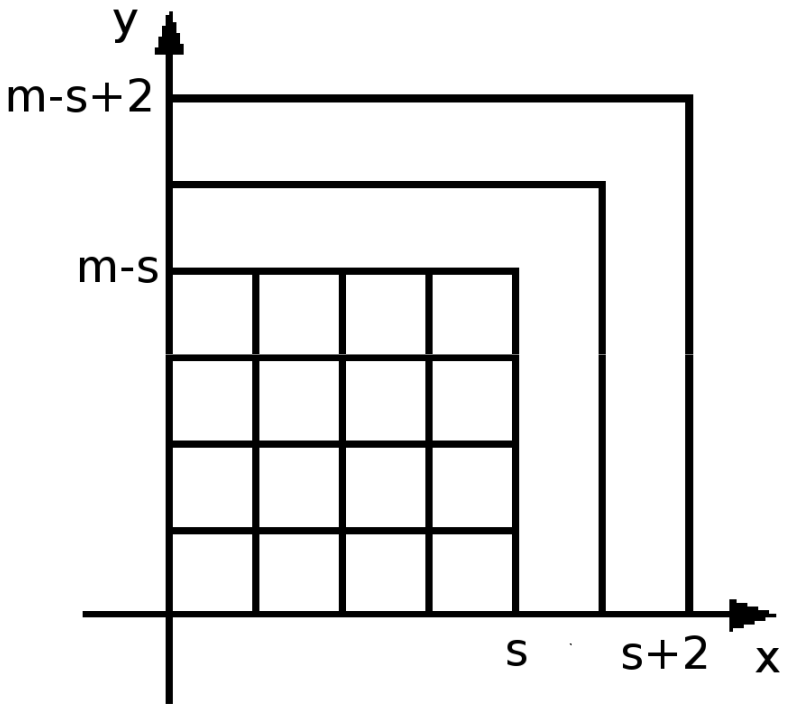
Sequential representation



Suppose that ***s*** is a number of units of memory allocated to **the first queue**, then (***m*−*s***) — number of units of memory allocated to **the second queue**.

The purpose of this research is to determine **the optimal allocation of memory between queues** (***s***), where the optimality criterion is the minimum average share of lost (by the **overflow**) elements of queues.

Sequential representation

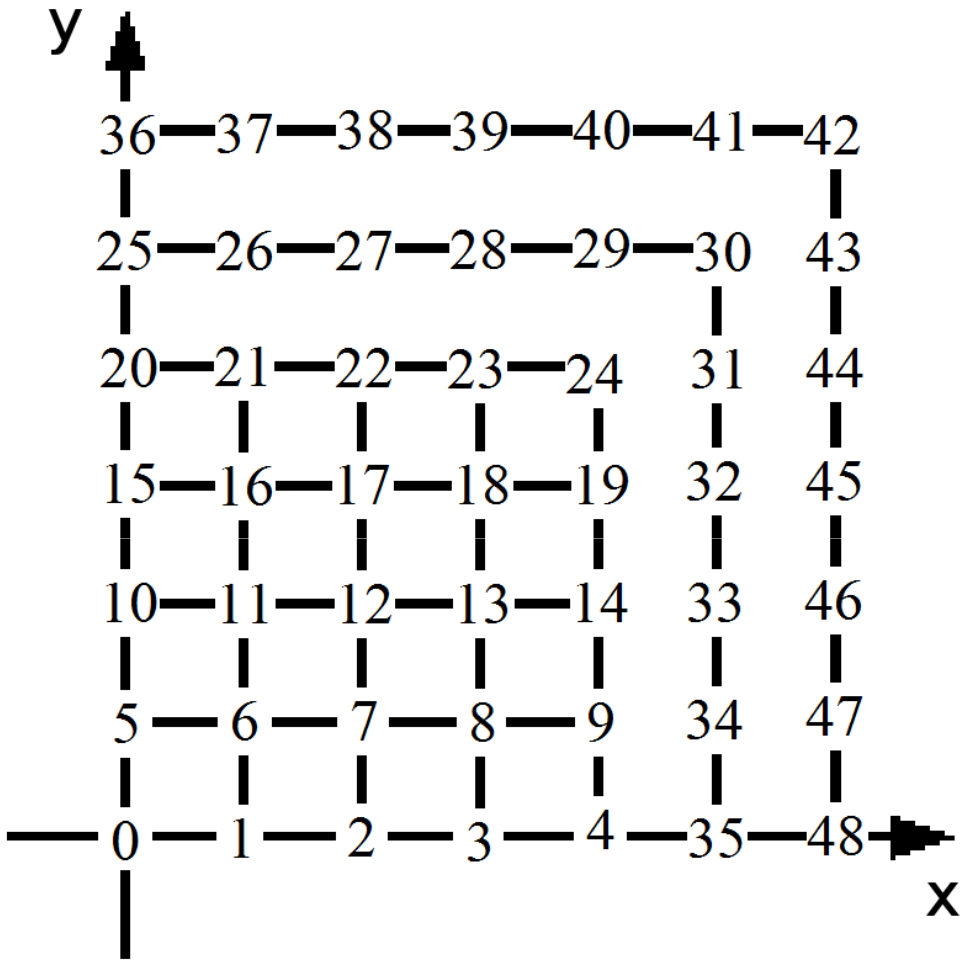


x and y are current lengths of **the first** and **the second queues**, respectively.

Consider **random walks** on two-dimensional space on an integer lattice in the area $0 \le x \le s+2$, $0 \le y \le m-s+2$.

Lines $x = s+1$ and $y = m-s+1$ form **the first reflective screen**; lines $x = s+2$, $y = m-s+2$ form **the second reflective screen**.

Sequential representation



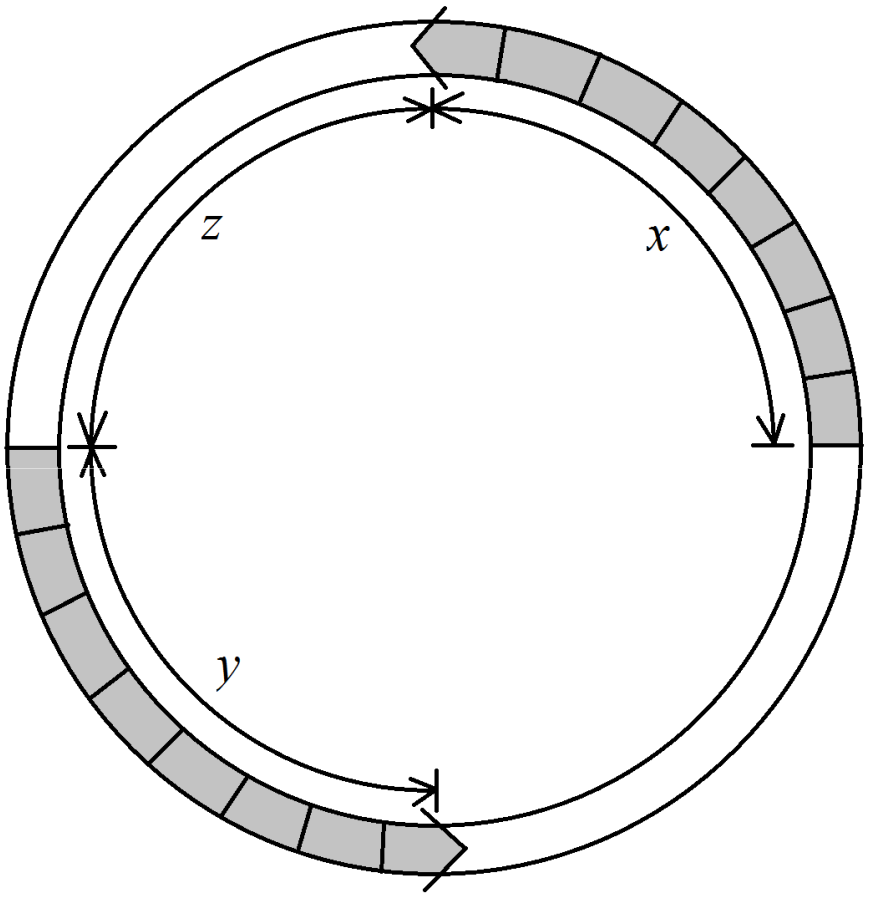
m = 8

s = 4

$$n = (m-s+3)*(s+3)$$

RuFiDiM 2014

One after another

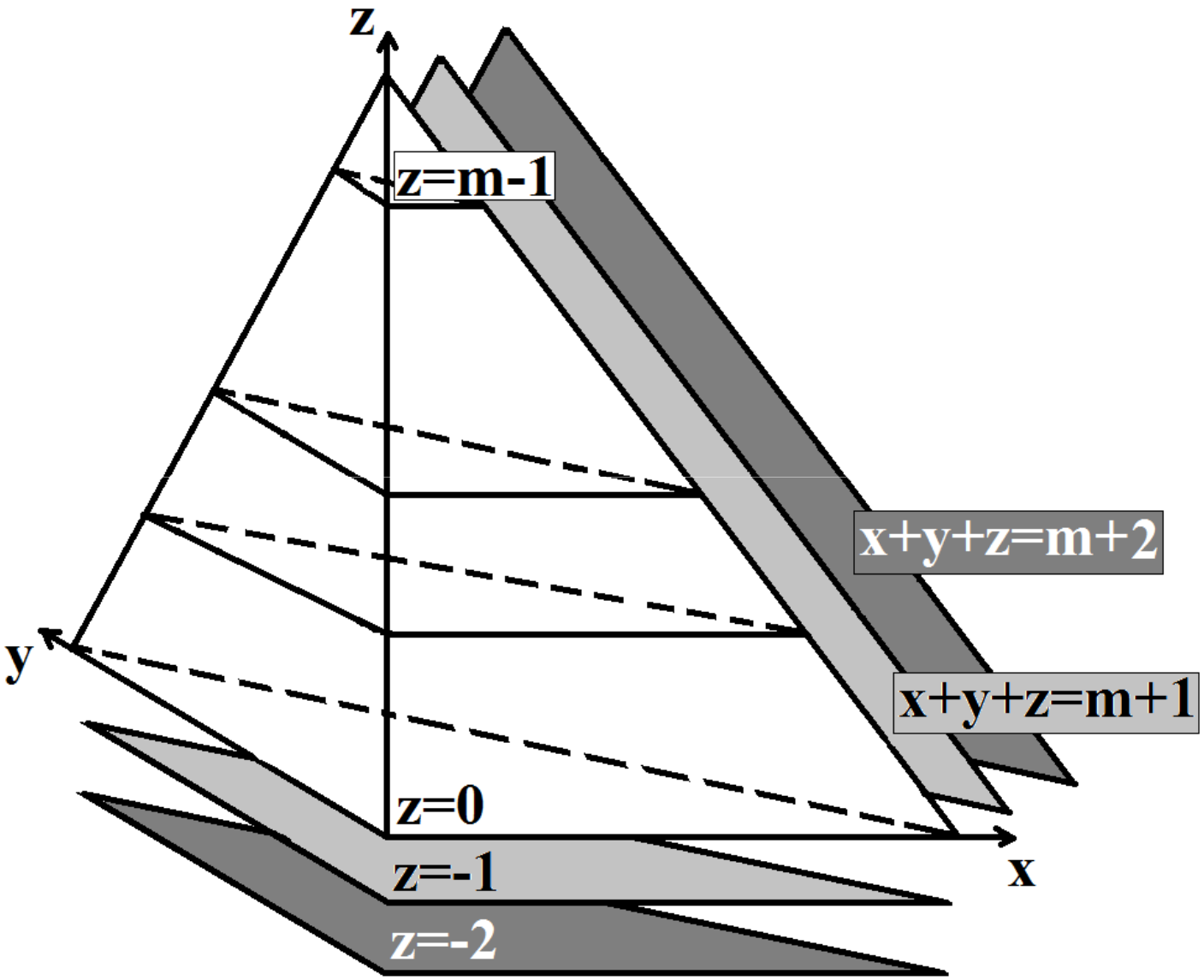


Assume, that x and y are current lengths of queues,

z — distance between the end of **the first queue** and the beginning of **the second**.

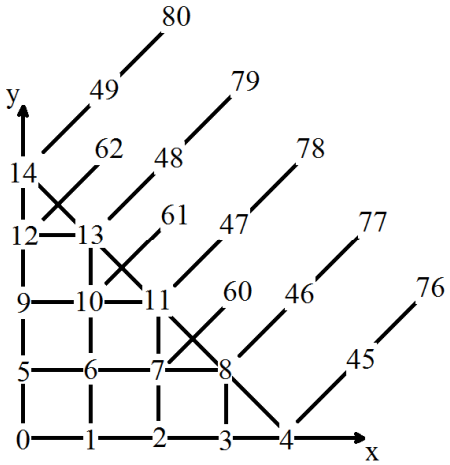
Aim of the research is to determine **the average share of lost elements** for comparison with the average share of lost elements with **sequential work** of parallel queues in the case of **the optimal partition of shared memory**.

One after another

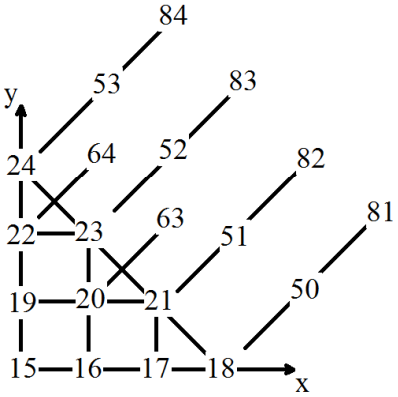


RuFiDiM 2014

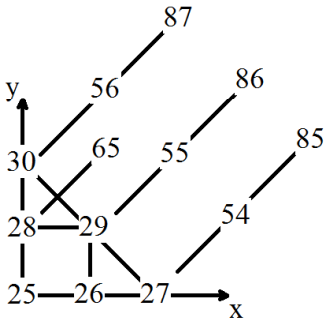
One after another



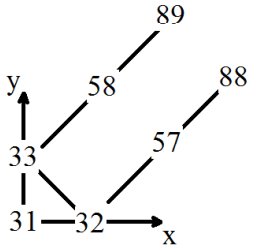
z = 0



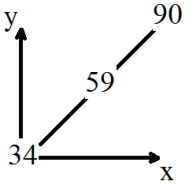
z = 1



z = 2

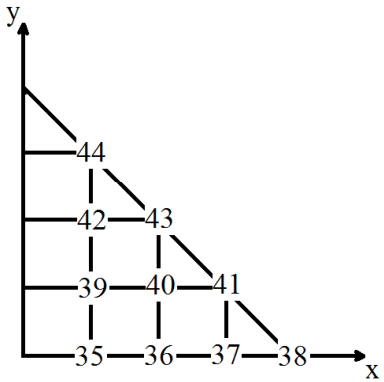


z = 3

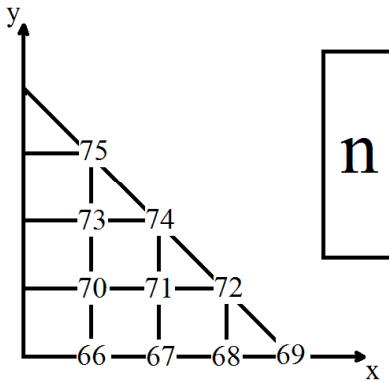


z = 4

m = 4



z = -1



z = -2

$$n = \sum_{i=0}^{m} \frac{(i+1)\cdot(i+2)}{2} + \frac{5m^2+7m+4}{2}$$

RuFiDiM 2014

Comparison of losses

Input	Value of the losses on overflow (m=12)		
	Optimal partition	Partition in half ($s = 6$)	One after another
$p_1 = 0.25, p_2 = 0.25, p_{12} = 0.25,$ $r_1 = 0.25$ $q_1 = 0.25, q_2 = 0.25, q_{12} = 0.25,$ $r_2 = 0.25$	0.075 ($s = 6$)	0.075	0.081
$p_1 = 0.35, p_2 = 0.20, p_{12} = 0.10,$ $r_1 = 0.35$ $q_1 = 0.25, q_2 = 0.25, q_{12} = 0.25,$ $r_2 = 0.25$	0.015 ($s = 8$)	0.019	0.016
$p_1 = 0.40, p_2 = 0.10, p_{12} = 0.10,$ $r_1 = 0.40$ $q_1 = 0.25, q_2 = 0.25, q_{12} = 0.25,$ $r_2 = 0.25$	0.028 ($s = 9$)	0.038	0.029
$p_1 = 0.425, p_2 = 0.05, p_{12} = 0.10,$ $r_1 = 0.425$ $q_1 = 0.25, q_2 = 0.25, q_{12} = 0.25,$ $r_2 = 0.25$	0.041 ($s = 9$)	0.052	0.040

All results were confirmed by simulation modeling.

Work stealing

In many implementations of parallel programming languages and libraries (Intel TBB, Cilk/Cilk Plus, Microsoft TPL, fork-join API for Java, et al) there is a task scheduler, where the working thread takes a new job to run from the end of its queue on the principle of LIFO. However, if the process (CPU) is idle, it is looking for a job in the queue of another process (CPU), and extracts it on the basis of FIFO

Thank you!

RuFiDiM 2014