Online Scheduling of Continuous Media Streams*

B. Monien, P. Berenbrink, R. Lüling, and M. Riedel**

University of Paderborn, Germany
E-mail: bm,pebe,rl,barcom@uni-paderborn.de

Abstract. We present a model for an interactive continuous media server. Such server systems work in an online environment without knowing future requirements of the customers. We apply competitive analysis to study different scenarios of the design and control of a server and we give upper and lower bounds for the competitive ratio.

1 Introduction

To provide high bandwidth interactive media to large number of customers suitable network and server technologies have to be available. Whereas the delivery of data without real time constraints like texts and images has been successfully solved (e.g. WWW service on Internet), this is not true for the delivery of real time continuous media streams in large networks. So one of the key elements in such an environment is the development of an interactive continuous media server (ICMS).

To store huge amounts of data and to allow simultaneous access for many customers a set of memory modules is necessary, e.g. a large array of hard disks. Our model consists of such a set of memory modules and a set of links. These two sets are completely connected. The customers get their data via the links and included buffers.

In our research we are interested in supporting decisions about memory bandwidth and storage policies of the data, and we study how the throughput is effected by allowing delays.

An ICMS works online, i.e. prospective requirements of the customers are unknown or uncertain because the customers have the freedom to access and to leave the system as well as to interrupt the transmission and to continue at any point of the media stream. However, the decisions about the current usage of the server resources have to be made immediately and a later change is impossible. We investigate the loss of performance due to this online information regime to see the consequences of different server constructions. To do so, we

* This work was partly supported by the EC ACTS project SICMA (Scalable Interactive Continuous Media Server – Design and Application), No. AC071, http://www.uni-paderborn.de/cs/sicma

** Supported by DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", GRK 124/2-96.

apply competitive analysis, introduced in [10]. It determines the maximal ratio between online and optimal offline solutions over all possible inputs.

In fact we focus on a very special scheduling problem. The memory modules are our resources and the data requests of the customers are the "jobs". The jobs can be managed by that subset of the resources storing a copy of the requested data, and in order to fulfill the real time constraints jobs have deadlines.

The literature about scheduling theory is extremely rich. A good introduction is [4] and [2], which also deals with resource constraints. [9] gives a survey about up to date online scheduling research whereas online interval scheduling with fixed start and end times is investigated in [8, 11].

The design of ICMS is a very active area of worldwide theoretical and experimental research (e.g. see [1, 3, 5, 6]). The work presented in this paper has been accomplished in the context of the SICMA project. This project, funded by the commission of the European community, aims at developing a parallel ICMS. The integration of sophisticated scheduling algorithms into the architecture of this server is one of the major aims of the SICMA project.

The material is organized as follows. Sec. 2 presents our model and Sec. 3 gives an overview of the results with some ideas of the proofs. In Sec. 4 we concentrate our view on a special model instance and prove a nontrivial upper bound. Finally, we mention a few open problems in Sec. 5.

2 The Model

Our model has three main parameters: b, c and d. These parameters describe the memory bandwidth (b), the number of data copies (c) and a tolerable delay (d).

In our model, $M := \{m_1, m_2, \ldots, m_n\}$ is the set of n memory modules. The data streams are split into packets of equal size. To relax access conflicts on memory modules each of these packets is located in c different memory modules. Our model works in discrete time steps and a memory module has the bandwidth to deliver up to b data packets in one step.

A customer requests at most one data packet per time step. Thus, we bound the number of concurrent customers by the total memory bandwidth bn and they are described by a set $R := \{\chi_1, \chi_2, \ldots, \chi_{bn}\}$. In the model we assume that each customer requests one arbitrary data packet, which means an access to one out of c memory modules storing the requested data, or does nothing at each time step. For our model the input is a request function $r : R \times \{1, \ldots, T\} \to M_c$, where $\{1, \ldots, T\}$ is the set of time steps and $M_c := \{A \mid A \subset M, |A| = c\} \cup \{\emptyset\}$. The absence of customers can be expressed with consecutive empty requests.

A request can be served immediately or with a maximal delay of d time steps.

We investigate two variants of the model. In the first one an algorithm has to construct a scheduling function $s : R \times \{1, \ldots, T\} \to (M \times T) \cup \{\emptyset\}$, which assigns a memory module $m \in M$ and a time of delivery $t \in \{1, \ldots, T\}$ for each request or assigns $\emptyset$ when the request is empty or it is unsatisfiable. We will call this problem "data access problem" (DAP).

In the second variant, which is a restriction of the first one, a customer, or his assigned link respectively, can only receive one data packet per time step. This model is more realistic and the results show a fundamental difference in the reachable online performance. This second variant is called "restricted DAP" (RDAP). Algorithms solving DAP or RDAP will be compared with the optimal offline algorithm OPT.

The model cannot guarantee for delivering the requested data packets[3]. Nevertheless, we want to serve as many requests as possible. So the performance of an algorithm ALG for DAP or RDAP under an input r is the number of successful requests, i.e. the number of delivered data packets, and it is denoted by $\mathbf{P}_{\mathrm{ALG}}(r)$. This objective function follows ideas of [8, 11] where no benefit is paid when a job violates its deadline.

Following the well known definition of competitiveness, we say an online algorithm $\mathcal{A}$ is ρ-competitive if there is a constant α such that for all inputs r

$$\mathbf{P}_{\mathrm{OPT}}(r) \leqq \rho\, \mathbf{P}_{\mathcal{A}}(r) + \alpha \quad .$$

The competitive ratio is then the infimum over all values ρ such that $\mathcal{A}$ is ρ-competitive.

In the next sections we study the influence of the three main parameters memory bandwidth b, number of copies c, and delay d to the competitive ratio for *deterministic* algorithms. We are interested in understanding the reachable benefit that arises from allowing larger delay, increased memory bandwidth, and number of data copies.

3 Results

The table below summarizes our current results. A competitive ratio of "1" indicates that the online algorithm reaches the same performance as an optimal offline algorithm while "> 1" expresses that no such online algorithm exists. In this work $\mathbb{N}$ denotes the natural numbers without 0.

problem	parameters	competitive ratio	
		lower bound	upper bound
DAP	$d = 0,\ b,\ c \in \mathbb{N}$		1
	$c = 1,\ b,\ d \in \mathbb{N}$		1
	$c > 1,\ b,\ d \in \mathbb{N}$	> 1	$\min\{2, n/c\}$
RDAP	$b,\ c,\ d \in \mathbb{N}$	> 1	$\min\{d+1, n/c\}$
	$b = c = d = 1$	$4/3$	$5/3$
	$c = d = 1,\ b \in \mathbb{N}$		$5/3$

The proof of the $5/3$ upper bound for RDAP is presented in Sec. 4. Below we sketch some of the proofs for the other results.

[3] Let us imagine requests of all customers to a few modules. We get "hot spots" and any algorithm can satisfy only a small subset of requests.

Upper Bounds Using a Matching Technique. First we describe the matching technique and afterwards we sketch the proofs for the upper bounds of rows one, three, and four in the above table.

The DAP can be modelled as a matching problem in a bipartite graph G with weighted nodes. Each request is represented as a node in U and has weight 1 (*request nodes*). Each resource, i.e. each memory module at each time step, is represented as a node with weight b in V (*resource nodes*) and in a matching such a node can be incident to maximal b matching edges. The set of edges E connects every non-empty request node to all resource nodes which are able to serve the request, i.e. to all c specified modules and all possible $d+1$ time steps. The capability of the memory modules is the only limiting factor, so every maximum matching in G describes an optimal solution for DAP. It is well known that a maximum matching can be constructed efficiently by an offline algorithm.

In the online version of DAP with no delay ($d = 0$) all dependencies between two time steps disappear. In fact every time step represents an independent offline problem and the optimal offline algorithm can solve it itself. Thus we get the first upper bound of 1.

For the DAP with parameters $d \geqq 1$, $c, b \in \mathbb{N}$ we have to construct the matching in G online. Karp *et al.* proved in [7] that 2 is an upper bound for the competitive ratio of the online bipartite matching for deterministic algorithms. This upper bound shown by the greedy algorithm also holds for the matching problem in G.

The upper bounds of $d+1$ and n/c are shown by an algorithm without using delays. At each time step it applies the matching technique to determine a maximum number of the current requests for immediate serving. All other requests are ignored. This algorithm does not violate the additional restriction of RDAP.

We get the $d+1$ upper bound for RDAP by the following observation. Due to the maximization of the above algorithm, no other algorithm (including OPT) can serve a higher number of the current requests in any of the $d+1$ possible time steps. The online algorithm uses only the first of them and an application of this fact for every time step gives the upper bound for the competitive ratio.

The n/c bound can be shown by another observation: If the current time step has no more than bc requests, then all of them will be served. Otherwise at least bc requests get their data packets because at least c memory modules are able to serve up to b requests. In our model, however, at most bn concurrent requests exist per time step and the optimal offline algorithm OPT can serve, at best, all of them. This proves the upper bound of n/c for DAP and RDAP. Here, a phenomenon of small system sizes is depicted: The competitive ratio decreases when the number of memory modules n (and therewith the maximal number of customers) becomes too small.

An Upper Bound Using Earliest Deadline First Heuristics. In the DAP with only one copy of every data packet ($c = 1$) all dependencies between memory

modules disappear. This problem is equivalent to the online uniprocessor deadline scheduling of unit time tasks which can be solved as optimal as in the offline case using the earliest deadline first heuristics. This gives the 1-competitiveness.

Lower Bounds. The nonexistence of 1-competitive online algorithms for DAP and RDAP with parameters $d \geqq 1$, $b \in \mathbb{N}$ and $c > 1$ (DAP), $c \in \mathbb{N}$ (RDAP) respectively, can be shown by an adversary argument. It works in two phases and the second phase depends on the decisions made by the online algorithm which are completely known to the adversary in advance.

In the first phase at time $t = 1$ the adversary constructs a request sequence in such a way that after d time steps all but one arbitrary request can be served. Thereafter, the system is in one of at least two different structured but equally valued configurations, depending on the request chosen by the algorithm for maximal delay. In the second phase the adversary continues the request sequence at time $t = d+1$ in a way that the system can just manage to serve all requests. However, due to the delayed request, which needs the same resource, the online algorithm cannot satisfy all of them.

The optimal offline algorithm is able to fulfill each request by choosing another system configuration before time $t = d + 1$. Then the maximal delayed request uses a resource not needed for serving the requests of the second phase.

Repeating the whole request sequence over time shows that no online algorithm can be 1-competitive.

The lower bound of $4/3$ for RDAP with parameters $b = c = d = 1$ can be shown by a similar strategy which appends new requests in a continuous way.

4 An Improved Upper Bound

Now we shift focus on a special model instance, the RDAP, where all three main parameters are equal to one. A memory module can deliver only one data packet per time step ($b = 1$), every data packet is stored only once in the system ($c = 1$), and a request has to be served immediately or in the next time step ($d = 1$). This is the model with smallest parameters for which the construction of 1-competitive algorithms is impossible.

The online algorithm $\mathcal{H}$ employed to show the $5/3$ upper bound needs a global view onto the system and works as follows:

for every time step $t \in \{1, \ldots, T\}$:
  read all incoming requests of time t and mark all of them corresponding to customers with a delayed request of time $t - 1$;
  for each memory module $m \in M$:
    phase 1: if there is a delayed request of time $t - 1$ on m
      then serve it
      else serve an unmarked request
    phase 2: delay one of the remaining requests for serving at time $t + 1$

Theorem. *$\mathcal{H}$ is $5/3$-competitive.*

Sketch of Proof. At first we need a few definitions and notations for a fixed input:

$\mathbf{P}_{\mathrm{ALG}}(T)$ – performance of algorithm ALG, i.e. number of served requests from the beginning up to time T.

$\mathbf{p}_m^{\mathrm{ALG}}(t)$ – local performance of ALG, i.e. number of served requests on memory module m at time t

$$\mathbf{P}_{\mathrm{ALG}}(T) = \sum_{t=1}^{T} \sum_{m \in M} \mathbf{p}_m^{\mathrm{ALG}}(t) \ .$$

$d_m^{\mathrm{ALG}}(t)$ – number of requests at time $t-1$ which are served by ALG at time t from memory module m.

$\varphi_m(t)$ – a local potential function on memory module m at time t which is defined as

$$\varphi_m(t) := \begin{cases} 0 \text{ , if } d_m^{\mathrm{OPT}}(t) = 0 \text{ and } d_m^{\mathcal{H}}(t) = 0 \\ 1 \text{ , if } d_m^{\mathrm{OPT}}(t) = 0 \text{ and } d_m^{\mathcal{H}}(t) = 1 \\ -1 \text{ , if } d_m^{\mathrm{OPT}}(t) = 1 \text{ and } d_m^{\mathcal{H}}(t) = 0 \\ \frac{2}{3} \text{ , if } d_m^{\mathrm{OPT}}(t) = 1 \text{ and } d_m^{\mathcal{H}}(t) = 1 \end{cases} .$$

$\Phi(t)$ – the global potential function which is defined as

$$\Phi(t) := \sum_{m \in M} \varphi_m(t) \ .$$

From this definition follows $\Phi(t) \in [-n, n]$.

In order to show the competitive ratio of $5/3$ it is sufficient to show

$$\mathbf{P}_{\mathrm{OPT}}(T) \leqq \frac{5}{3} \mathbf{P}_{\mathcal{H}}(T) + \Phi(T+1) \tag{1}$$

which is equivalent (using the above definitions and the fact that at the starting time it holds for each $m \in M : \ d_m^{\mathrm{OPT}}(1) = d_m^{\mathcal{H}}(1) = 0 \ \Rightarrow \ \varphi_m(1) = 0$) to:

$$0 \leqq \sum_{t=1}^{T} \sum_{m \in M} \left[\frac{5}{3} \mathbf{p}_m^{\mathcal{H}}(t) - \mathbf{p}_m^{\mathrm{OPT}}(t) + \varphi_m(t+1) - \varphi_m(t) \right] \ . \tag{2}$$

Thus, it is sufficient to concentrate our view on local situations where a local situation is defined by a memory module $m \in M$ at a time t, $1 \leqq t \leqq T$. For reasons of convenience we define the term for the value of such a local situation in (2) as $\ell_m(t)$:

$$\ell_m(t) := \frac{5}{3} \mathbf{p}_m^{\mathcal{H}}(t) - \mathbf{p}_m^{\mathrm{OPT}}(t) + \varphi_m(t+1) - \varphi_m(t) \ .$$

Equation (2) and the terms $\ell_m(t)$ compare an arbitrary but fixed optimal solution of OPT with the online solution of $\mathcal{H}$. If we consider all feasible combinations of the variables in such a local term $\ell_m(t)$ (based on the input and the decisions made by OPT and $\mathcal{H}$), we can observe that $\ell_m(t) \geqq \frac{1}{3}$ in all but three cases. These exceptions are:

Case 1: There are at least two requests to the memory module m, all of them are marked by $\mathcal{H}$. OPT and $\mathcal{H}$ have no delayed requests of time $t-1$ ($d_m^{\mathrm{OPT}}(t) = d_m^{\mathcal{H}}(t) = 0$). OPT serves one request ($\mathbf{p}_m^{\mathrm{OPT}}(t) = 1$) and will serve another one with delay ($d_m^{\mathrm{OPT}}(t+1) = 1$).
Under this condition $\mathcal{H}$ delivers no packet ($\mathbf{p}_m^{\mathcal{H}}(t) = 0$) and will serve one request with delay ($d_m^{\mathcal{H}}(t+1) = 1$); therefore $\ell_m(t) = -\frac{1}{3}$.

Case 2: Like Case 1 but algorithm OPT will not serve a request with delay ($d_m^{\mathrm{OPT}}(t+1) = 0$); therefore $\ell_m(t) = \pm 0$.

Case 3: There is one request to the memory module m which is marked by $\mathcal{H}$ and no delayed requests ($d_m^{\mathcal{H}}(t) = d_m^{\mathrm{OPT}}(t) = 0$). OPT serves this request at time t ($\mathbf{p}_m^{\mathrm{OPT}}(t) = 1$).
Under this condition $d_m^{\mathrm{OPT}}(t+1) = 0$ and $\mathcal{H}$ has to serve this request with delay ($\mathbf{p}_m^{\mathcal{H}}(t) = 0$, $d_m^{\mathcal{H}}(t+1) = 1$); therefore $\ell_m(t) = \pm 0$.

In Case 1 the term $\ell_m(t)$ is negative but such a situation needs special preconditions. So we will prove the existence of a set of local situations including all of Case 1 with a non negative overall sum of their terms $\ell_m(t)$.

In a Case 1-situation at time t all requests are marked. Therefore, we can uniquely identify at least two local situations at memory modules m' and m'' occurring one time step before where $\mathcal{H}$ delayed exact one request of the current customers at each module ($d_{m'}^{\mathcal{H}}(t) = d_{m''}^{\mathcal{H}}(t) = 1$). For these local situations can hold $\ell_{m'}(t-1) \geqq \frac{1}{3}$ or $\ell_{m''}(t-1) \geqq \frac{1}{3}$. Then the negative value is equalized. However, it is also possible that both situations are of Case 1, 2 or 3. In all three cases $\mathcal{H}$ cannot deliver a data packet because all requests are marked. Applying the same argument we can identify one (Case 3) or at least two (Case 1 and 2) predecessor situations.

For every Case 1-situation at time $t' \leqq T$ we can construct a *dependency tree* where all leaves have a local value $\ell_m(t) \geqq \frac{1}{3}$. Thereby, other dependency trees are possibly absorbed. At starting time no delayed request exists, so no situations of Case 1, 2 or 3 can occur. This fact bounds the depth of the tree by t'.

This tree has more leaves ($\ell_m(t) \geqq +\frac{1}{3}$) than inner nodes with at least two predecessors (possibly of Case 1; $\ell_m(t) = -\frac{1}{3}$) and the overall sum of the values $\ell_m(t)$ of the situations described by such a tree is non negative. Comparing the solutions of $\mathcal{H}$ and OPT it is possible to find a forest of such dependency trees, which includes all situations of Case 1, and the sum of all the local terms $\ell_m(t)$ is non negative with respect to inequality (2).

□

A careful inspection of the proof gives us an adversary showing that this $5/3$ bound is tight for algorithm $\mathcal{H}$.

This algorithm and the proof can be extended for arbitrary memory bandwidth $b \in \mathbb{N}$ in a straight forward manner and the same upper bound holds.

5 Open Problems

The bounds for the competitive ratio of the introduced model still have gaps. E.g. algorithm $\mathcal{H}$ makes the decisions about delaying requests in phase 2 too

early. We have the feeling that it should be possible to improve the upper bound by using an algorithm waiting on the new requests before deciding which ones are served with delay. But up to now we have not been able to prove it.

Also, we like to extend the restrictions to real network structures[4] between the memory modules and the customers. A very interesting question is the influence of an additional lookahead while the buffers are bounded by a small constant.

If convincing motivated assumptions or knowledge about the input distribution are available, it is necessary to develop and study functions to map the data to memory modules supporting this input distribution.

Finally, it would be interesting to know whether randomized online algorithms can essentially help to decrease the bounds.

References

1. S. Aggarwal, J. A. Garay, and A. Herzberg. Adaptive video on demand. In P. G. Spirakis, editor, *Proceedings of the Third Annual European Symposium on Algorithms*, LNCS 979, pages 538–553, Berlin-Heidelberg-New York, 1995. Springer-Verlag.
2. J. Błażewicz, W. Cellary, R. Słowiński, and J. Węglarz. *Scheduling under Resource Constraints - Deterministic Models*, volume 7 of *Annals of Oaeration Research*. J.C. Baltzer AG, Scientific Publishing Company, Basel, 1986.
3. C. Bouras, V. Kapoulas, G. E. Pantziou, and P. G. Spirakis. Competitive scheduling schemes for video on demand. Technical Report CTI-TR 96.5.12, Computer Technology Institute, Patras, Greece, 1996.
4. P. Brucker. *Scheduling Algorithms*. Springer-Verlag, Berlin-Heidelberg-New York, 1995.
5. A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and caching in large-scale video servers. In *Proceedings of COMPCON 1995*, pages 217–224, 1995.
6. D. J. Gemmell, H. M. Vin, D. D. Kandur, P. V. Rangan, and L. Rowe. Multimedia storage servers: A tutorial. *IEEE Computer*, 28(5):40–49, May 1995.
7. R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.
8. R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 302–311, 1994.
9. J. Sgall. On-line scheduling — a survey. In A. Fiat and G. J. Woeginger, editors, *Dagstuhl Seminar on On-Line Algorithms (June 24–28, 1996)*, to appear in LNCS in Summer 1997. Springer-Verlag, Berlin-Heidelberg-New York.
10. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, Feb. 1985.
11. G. J. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.

This article was processed using the LaTeX macro package with LLNCS style

[4] networks with nodes of constant and small degree