Online request scheduling subject to a percentile response time SLA in a distributed cloud

Keerthana Boloor∗†, Rada Chirkova†‡, Timo Salo†, and Yannis Viniotis∗‡

∗Department of Electrical and Computer Engineering, NC State University, Raleigh, North Carolina 27695
†Department of Computer Science, NC State University, Raleigh, North Carolina 27695
‡IBM Software Group, RTP, North Carolina 27709

kboolor@ncsu.edu, chirkova@csc.ncsu.edu, tjsalo@us.ibm.com, candice@ncsu.edu

Abstract—We consider geographically distributed data centers forming a collectively managed cloud computing system hosting multiple applications, each subject to Service Level Agreements (SLA). The Service Level Agreements for each application require the response time of a certain percentile of the input requests to be less than a specified value, with the non-conforming requests being charged a penalty. We present a novel approach of heuristics based request scheduling at each server in each of the geographically distributed data centers, to globally minimize the penalty charged to the cloud computing system. We evaluate two variants of our heuristic-based approach, one based on the simulated annealing method of neighborhood searches and another based on gi-FIFO scheduling, which has been analytically proven to be the best schedule for percentile goals in a single machine, multi-class problem. We also compare our approaches with FIFO scheduling.

I. Introduction and motivation

Deployment of on-demand e-business applications in large cloud computing systems is getting increasingly pervasive. The significant reduction of the total cost of ownership by deploying web-based applications in massive data centers is resulting in many businesses opting to host their applications in a cloud. The e-business applications with implementations usually based on service oriented architectures tend to be global in scale and require deployments in geographically distributed data centers for scalability and survivability [1] [2].

Most enterprise applications hosted by a cloud computing system provider are associated with a Service Level Agreement (SLA), which specifies terms and conditions of the service provided by the cloud for the application [3]. With the applications deployed across multiple data centers, there is a need for resource allocation and request scheduling techniques for the satisfaction of the SLA of each application, globally, across the geographically distributed data centers.

Typically, SLAs for business applications specify (among other constraints) certain guarantees in terms of the fraction of requests serviced, as opposed to average-performance criterion. Thus, many service level agreements are designed to provide specific percentile-based performance goals. Recent business trends in cloud computing systems have shown increasing adoption of fixed-step percentile SLAs, where a certain fraction of service requests for a hosted application is required to have a specific response time [4]. As geographically distributed data centers, each having a large number of servers, form a cloud computing system, this percentile SLA has to be respected globally across all the servers among all the data centers.

In this paper we consider the single-step percentile SLA, where the fraction of service requests to be executed within a certain response time is specified, along with the penalty charged on the cloud on the non-conformance of the percentile requirement. The formal description of the SLA we consider is as follows: Let \(X\%\) be the fraction of requests of a particular application which need to have a response time less than \(r\) seconds. If the percentile of requests that have response time less than \(r\) seconds is less than \(X\%\), then each of the non-conforming requests contributing to the drop in the percentile is charged a penalty of \(P\$, as shown in Fig. 1. In our paper we consider the problem where a cloud computing system consists of multiple geographically distributed data centers, each with a large number of servers. The centers host collectively multiple classes of applications, which are each negotiated with a single step-wise SLA and so have to be adhered to globally by the cloud. We consider the situation where requests for different applications arrive at the servers hosting the applications, and the requests queued at each server have to be scheduled optimally in order to minimize the penalty charged according to the SLAs negotiated. We make no assumptions about any prior knowledge about the number of requests arriving at the cloud computing systems for different applications or at different data centers. Hence there is a need for a dynamic scheduling algorithm, which would schedule incoming requests at each server, taking into
consideration the global conformance of the percentile single-step SLAs for all classes of applications.

In this paper we propose a novel Online algorithm for the scheduling of requests at the end servers of the data centers of a cloud. We perform extensive evaluations to demonstrate that the algorithm provides optimum schedules that globally minimize the total penalty in the cloud. To the best of our knowledge, this is the first effort towards considering global conformance of percentile SLAs and utilizing an Online approach for scheduling of service requests at each of the end servers to collectively minimize the penalty.

Resource management techniques for cloud computing systems have been researched extensively. However, none so far have dealt with request scheduling in geographically distributed data centers or with global conformance to percentile SLAs. In [3] the authors derive a closed-form expression for average response time in terms of scheduling and routing of requests for a single data center. They use tabu search for optimum solutions based on a step-wise SLA, with penalties allotted for steps of response time and not percentiles. Authors of [4] also consider step-wise percentile SLAs and propose scheduling algorithms for a single database server; in this work we propose a distributed solution. Authors of [5] provide an analytical solution for resource optimization subject to percentile response time, by modeling the system as an overtake-free open tandem queuing network with feedback. They provide closed-form expressions of the probability distribution function of the response time. In contrast, we provide a heuristic-based scheduling algorithm for global conformance in a distributed data center topology.

In summary, our contributions in this paper are as follows:

- We identify the need for SLA-based, penalty-minimizing request scheduling at end servers of a cloud computing system with geographically distributed data centers.
- We propose an Online request scheduling algorithm for geographically distributed data centers hosting multiple classes of requests, aiming to minimize the penalty charged on the cloud computing system.
- We propose and evaluate two heuristic-based variants of our algorithm, one based on Simulated Annealing [6] and another based on the gi-FIFO schedule, which has been mathematically proven in [7] to be the most suitable for percentile SLAs for a single server serving multiple classes.

The paper is organized as follows. In Section II, we describe the topology of the cloud computing system under consideration. In Section III, we explain our heuristic-based scheduling algorithm. In Section IV, we evaluate our algorithm and compare it with alternatives.

II. Problem formulation

A. System model

The general architecture of the system is shown in Fig. 2. The following are the key elements of the system:

- Clients. These are nodes that generate the service requests forwarded to the servers at the different data centers of the cloud. The clients are represented as the internet cloud in Fig. 2.

- Data centers and hosted applications. A data center is a cluster of a large number of networked computing resources. In the topology considered, multiple geographically distributed data centers form the cloud computing system with each data center hosting the same set of applications as shown in Fig. 2. Each data center receives web service requests for applications from clients. An application’s web service end points are replicated in all the data centers, i.e., any data center is capable of serving a request for any application. Each application is identified by a class. So if the cloud hosts K applications, there are K classes of requests to be served by the data centers.

- Model of resources in a data center. Each data center has a number of servers (resources) for processing the web service requests. A server processes a single request to completion each time. A request being processed cannot be preempted. Requests arriving when the server is busy are queued. Each server in a data center can process a request of any class (application). So each server in the data center is modeled as serving multiple single-class queues, each queue holding requests of a particular class as shown in Fig 3.

- Percentile Service Level Agreements. In this problem we consider percentile SLAs where the percentile of service requests to be executed within a certain response time is given, along with the penalty charged on the cloud for the non-conformance of any service request beyond the stated percentile as shown in Fig. 1. The SLA is global in definition, i.e., all the data centers in the cloud have to collectively respect the SLA. So, the response time and
percentile constraints of the application should be met at
the global cloud system level.

The service requests for different applications from the clients
can be routed to any end server at any data center in the
cloud. The routing of service requests to the different servers
is based on cloud management policies depending on load
of individual servers, proximity to databases etc. (We do not
consider the problem of routing the service requests to the
servers.) The new requests are queued for scheduling at the
servers. The most common scheduling principle in a non-
preemptive system is the First-in-First-out (FIFO) policy. In
FIFO, some requests can be delayed beyond the contraints
specified in the SLA and incur penalties. Requests of different
classes when delayed, incur different penalties, based on the
current number of requests that conform to the response-
time constraint mentioned in the SLA. The local scheduling
policy at each end server should be such that the cloud
globally minimizes the penalty. There is a need for a dynamic
scheduling policy which schedules new requests at end servers,
to adhere to the SLAs specified for each class of service
request and thus minimizing the penalty incurred globally.

B. Problem statement

We want to schedule the incoming service requests of
different classes locally at the servers in the geographically
distributed data centers so as to minimize the total global
penalty incurred.

More specifically, we want to determine a scheduling algo-

\[ \min \sum_{1 \leq j \leq K} \text{pen}_j \]  

(1)

where \( \text{pen}_j \) is the penalty charged for non-conformance of the
requests of class \( j \) as described by Fig. 1 for the entire cloud.

III. Online scheduling algorithm for global percentile
SLA conformance

In this paper, we propose a distributed, measurement based
policy to schedule requests queued at individual servers lo-
cated in each data center in the cloud. There are two basic
ideas behind the proposed scheduling policy. The first is that,
for the scheduling at each server to be based on the current
global SLA conformance, we propose periodic updates of
conformance levels of each application between the geograph-
ically distributed data centers so that each data center is aware
of the current global conformance at periodic intervals. The
second is the calculation of penalty incurred by each arriving
request, which is charged, if it does not meet the response
time constraint; this calculation (see Algorithm 3) is done
adaptively, based on the current global non-conformance of
the class of the request, which is the fraction of requests of
the class which have not met the response time specified in the
SLA \((1 - cc_k)\). The aim is to ensure that incoming requests of
classes with higher current conformance with respect to their
SLA are assigned a lower penalty and vice versa.

1) Algorithm description: The observation interval \((T\) 
secs) during which the SLA has to be met is divided into
several subintervals, the number of which is configurable.
The observation interval is applicable to the entire distributed
cloud i.e the observation interval is the duration for which the
equation 1 has to be minimal. This can repeat indefinitely or
a set number of times configurable by the cloud administrator.
The subintervals start and end at the same instant in all the
data centers (synchronization). Each subinterval is partitioned
into a “scheduling phase” and an “adaptation phase”, as shown
in Fig. 4 and formalized in Algorithm 1. The two phases are
explained in detail in Algorithms 2 and 3.

Adaptation phase In the adaptation phase, described in Al-
gorithm 2, each data center exchanges its current conformance
levels of all classes with other data centers in the cloud, and
each data center calculates the updated conformance levels for
all the classes. This updated current conformance level \((cc_k\)
in Algorithm 2 and 3) is used in individual request-penalty
calculations during the scheduling phase.

Scheduling phase In the scheduling phase, run at each end-
server, shown in Algorithm 3, each arriving request at end-
server is assigned a penalty and scheduled. We calculate the
effect of delaying the recently arrived request on the current
non-conformance, so the numerator and denominator in equa-
tion (A) of Algorithm 3 are both incremented by one request. If
delaying this request causes the non-conformance to increase
beyond that given in the SLA, then the request is assigned
a penalty \(p_k\), else it is assigned a penalty of 0. This penalty
assignment \((pa_j)\) is performed at each end server, upon arrival
of every request and so each queued request has a penalty
assigned, which is charged if it does not meet the response
time. So our multi-class, multi-server, percentile penalty-based
scheduling problem is now converted to the well-investigated,
multi-class, single machine scheduling problem, in which
the penalty charged is dependent on the request completion
time [8] shown in Equation (B) of Algorithm 4. Determining
the schedule that minimizes the total penalty for the single
server in Equation (C) in Algorithm 4 is known to be NP-
hard [9]; typically, such a problem is solved with heuristics for
neighborhood searches [10]. The neighborhood search method
we have chosen is Simulated Annealing (SA) [6]. In this
iterative method, summarized in Algorithm 4, we begin with
a seed schedule in the first iteration, typically ordered on the
arrival instants (in our implementation); in each subsequent
iteration, we re-order the requests queued and calculate the
penalty for each schedule obtained. The next schedule to
move to is chosen in random and this is continued for a set
number of iterations. At the end, we choose the schedule
with the lowest penalty [6]. We investigate two methods
of neighborhood search in simulated annealing, namely last
insertion and pairwise interchange [10]; we have selected them
for the computational overhead they introduce. They differ in
the way of obtaining the next (neighbor) schedule. In pair-
wise interchange, we interchange the order of two randomly
selected service requests in each neighbor schedule and so we
have a maximum of \(n(n-1)/2\) number of schedules with
n being the number of requests queued. However, depending on the number of iterations, all the schedules may not occur. In last insertion, a new neighbor is generated by inserting the recently arrived request in different positions of the schedule leading to a total of \((n - 1)\) schedules.

A heuristic similar to simulated annealing is Tabu Search [10]. Tabu search prevents the occurrence of local optima in any neighborhood search. In tabu search, a search for the optimum solution is carried out in a similar manner as in simulated annealing, in addition, a tabu list is maintained which holds a configurable number of past traversed schedules. The currently found schedule is compared to the list and if it is found, it is discarded and a new schedule is obtained in its place. The two variants of neighborhood search proposed for simulated annealing can be utilized for Tabu search as well.

Our second method of scheduling the arriving request at each end server is the gi-FIFO policy [7], which is described as follows: First, choose the request class with the highest penalty; then, amongst all the queued requests of the chosen class, choose one with a maximum waiting time but which results in a response time less than or equal to \(r\). If no such request exists, choose the request with a higher waiting time resulting in a response time greater than \(r\).

The gi-FIFO policy was shown to maximize delay percentiles in single-server systems [7]. In Section IV, we compare the two variants of simulated annealing, Tabu search with pairwise interchange and gi-FIFO and FIFO policies with respect to minimizing penalty in our system.

2) Assumptions: In formulating the algorithm we have made the following assumptions:
- The latency in exchanging messages between the geographically distributed clusters is negligible when compared to the request inter-arrival and service times.
- The time required for updates of the status of an arriving request, to propagate to all servers in the data center is negligible compared to the service request inter-arrival and service times.
- The processing time of a service request at a server is known on its arrival.
- The data-centers are synchronized and so the subintervals start and end at the same instant at all data-centers.

![Fig. 4: Periodic scheduling and adaptation at each data center.](image-url)

### IV. Evaluation

In this paper, we center our evaluation (performed via simulations) on the following questions. The first question we investigate is “How does the solution algorithm suggested for solving the minimization problem in equation 1 perform?” A representative scenario involves a cloud computing system with (a) \(K = 10\) classes of services each with SLAs as described in Fig. 1, (b) \(N = 5\) geographically distributed data centers, (c) 10 servers in each data center, (d) the input arrival process is Poisson, (e) the service processes are exponential, uniform across all classes. Typical results for such simulation runs show that the simulated annealing algorithm substantially outperforms the FIFO policy. For example, Fig. 6 shows (with 95% confidence intervals) the penalty incurred in FIFO is much higher (many times, almost \(10^3\) times) than that incurred by our algorithm for varying input rates. Moreover as Fig. 5 shows (with 95% confidence intervals) Online scheduling typically outperforms FIFO even on a per class basis, with conformance levels for each class matching that required by

---

#### Algorithm 1 Online scheduling algorithm for global penalty minimization.

**Input:**
- Length of observation interval: \(T\)
- Number of subintervals: \(Z\)

**Output:** Minimization of Equation 1.

```plaintext
for z = 1 to Z do
  Scheduling phase as in Algorithm 3
  Adaptation phase as in Algorithm 2
end for
```

#### Algorithm 2 Adaptation phase at datacenter \(d\), calculating updated global conformance levels.

**Input:**
- Number of geographically distributed data centers: \(N\)
- Number of classes of service requests: \(K\)
- Number of requests serviced of class \(k\) in data center \(l\) in current subinterval: \(X_{slk}\) \(\forall k \in \{1, \ldots, K\}, \forall l \in \{1, \ldots, N\}\)
- Number of requests class \(k\) in data center \(l\) which met the required response time in current subinterval: \(X_{rlk}\)
- Total number of requests serviced of class \(k\) from the start of the observation interval as measured by \(d\): \(X_k\)
- Current conformance of class \(k\) in cloud as calculated by data center \(d\): \(cc_k\)

**Output:** Updated current global conformance calculated by \(d\):

\[ cc_k \forall k \in \{1, \ldots, K\} \]

```plaintext
for k = 1 to K do
  temp = 0
  temp2 = X_k
  for i = 1 to \((d - 1)\) and \(i = (d + 1)\) to \(N\) do
    temp = temp + X_{rlk}
    X_k = X_k + X_{slk} /*Get updates from all other data centers*/
  end for
  cc_k = \((cc_k \times temp2 + temp) / X_k\) /*Updated conformance level*/
end for
```
than classes with lower such values. So the third question:

Algorithm 3 During scheduling phase: A request $j$ arrives at endserver $s$ for service.

**Input:**
- Number of servers in each data center: $m_l \forall l \in \{1, \ldots, N\}$
- Request $j$ of class $k$ arrived at queue of server $s$ at
data center $l \forall k \in \{1, \ldots, K\}, \forall s \in \{1, \ldots, m_l\}, \forall l \in \{1, \ldots, N\}$
- Penalty per request of class $k$ on non-conformance as per SLA: $p_k \forall k \in \{1, \ldots, K\}$
- Required global conformance of class $k$ as per SLA: $c_k \forall k \in \{1, \ldots, K\}$
- Required response time conformance of class $k$ as per SLA: $r_k \forall k \in \{1, \ldots, K\}$
- Number of requests queued at server $s$ at data center $l$
at time $t$: $q_{ls}$

**Output:** Penalty applied for newly arrived request $j$ in the
schedule if it does not meet $r_k$: $pa_j$

if $q_{ls} = 0$ and server $s$ is free then
    Dispatch request $j$ for processing at end-server $s$
    Depending if request $j$ met the response time, update the
    conformance for class $k$.
else
    \[ nonconf = ((1 - cc_k) \times X_k + 1)/(X_k + 1) \]  \hspace{1cm} (A)
    if $nonconf > (1 - c_k)$ then
        $pa_j = p_k$ /*non-conformance high, $p_k$ penalty assigned to $j^\dagger$*/
    else
        $pa_j = 0$ /*non-conformance low, 0 penalty assigned to $j^\dagger$*/
    end if
    Insert request $j$ charged with penalty $pa_j$ in the queue
    of end server $s$
    Apply heuristics for scheduling the request as in Algo-
    rithm 4.
end if

we pose is “Does the Online scheduling algorithm favor
requests with higher penalty?” To answer this we simulated
the algorithm with the same cloud computing configuration as
mentioned before but with just two classes of service requests,
class one with penalty of 0.9$\$ for each non-conforming request
and class two with penalty of 0.1$\$ for each non-conforming
request, with a cut-off conformance of 90% and the same
response time requirement for both. The input request rates for
the two classes are the same. We ran the simulation multiple
times varying the input request rate each time and the results
in Fig. 8 show that the requests for class one are favoured
over requests of class two by our scheduling algorithm, thus
scheduling the requests with higher penalty ahead of requests
with lower penalty but FIFO scheduling favours requests of
both classes equally.

The scheduling algorithm at each geographically distributed
data center should aim to minimize the global penalty. This is

Algorithm 4 Simulated Annealing/gi-FIFO based optimum sched-
ule for requests queued at server.

**Input:**
- Number of requests queued at server $s$ at datacenter $l$
at time $t$: $n \forall s \in \{1, \ldots, m_l\}, \forall l \in \{1, \ldots, N\}$
- Process time of request $j$: $P_j \forall j \in \{1, \ldots, n\}$
- Starting time of request $j$: $x_j \forall j \in \{1, \ldots, n\}$
- Penalty charged if request $j$ is scheduled at time $x_j$:
    $\text{pen}_{ij}(x_j)$
- If scheduling the request $j$ of class $k$ at time $x_j$ causes
    the response time of $j$ to be less than or equal to $r_k$,
    then $\text{pen}_{i}(x_j) = 0$, otherwise $\text{pen}_{i}(x_j) = \text{pa}_j$ $\forall \text{pa}_j$ can be
    either 0 or $p_k$ depending on the current conformance of class $k$ as
    assigned in Algorithm 3$\dagger$.
- $X$: \{ $x$ | $x_{j1} = t$ and $x_{jz+1} = x_{jz} + P_{jz}, z = 1, \ldots, n$
    for some permutation $j_1, \ldots, j_n$ of $1, \ldots, n$ \} \hspace{1cm} (B)

We have to obtain optimum schedule of the $n$ requests
queued where $\min_{x \in X} \sum_{1 \leq j \leq n} \text{pen}_{i}(x_j)$

**Output:** Most optimum schedule of requests queued at server $s$

SA: Use current schedule as starting seed schedule
iterations = 0
repeat
    penalty = Compute penalty with current schedule
    new schedule = pairwise interchange or last insertion
    delta = Penalty of new schedule - penalty
    if delta < 0 then
        final penalty = penalty + delta
        Final schedule = new schedule
    end if
    current schedule = new schedule
    iterations ++
until iterations < MAXITERATION
OR
Apply gi-FIFO policy for the requests queued
achieved by the periodic adaptation phase as shown in Fig. 4. So “Is the algorithm distributed in nature?” is our next question. In the same configuration as in the first experiment, consider, for example, two clusters, one with large number of resources and another with small number of resources. Results in Fig. 9 show that the locally calculated current conformance $cc_k$ values, at data center with low resources increase after considering the global value.

The next question is centered around the approximating variants proposed: “Which variant of our algorithm (among last insertion, pairwise interchange simulated annealing, tabu search and gi-FIFO) obtains schedules with lower penalty?” Our simulations did not reveal a clear winner; in general, pairwise interchange performed better than the other two (and so we simulate only pairwise interchange variant of Tabu search). A typical result from our run is shown in Fig. 10. In this figure, we compute the total penalty as a function of request rates for both pairwise interchange and gi-FIFO with very stringent SLA criterion, and when the system is stressed where even at lower request rates, the penalty incurred is high. As we can see, at lower request rates (lesser stress), both gi-FIFO and pairwise interchange perform equally well; at higher request rates (more stress), pairwise interchange performs much better due to its almost exhaustive search for the optimum schedule. However, pairwise interchange takes significantly longer than gi-FIFO to execute and so when the system is less stressed (with comparatively lower rate of requests and more relaxed SLA constraints), gi-FIFO is as effective as pairwise interchange. Fig. 11 (obtained with the same topology configuration as in experiment one) depicts a typical per class behavior for all three algorithms: no algorithm meets all per class requirements and no algorithm is a consistent winner, on a per class basis. In last insertion, the penalty only depends on the position of the newly arrived request in the schedule and not on finding the best overall schedule as in the case of pairwise interchange. Also shown in Fig. 11 is that gi-FIFO exceeds pairwise interchange for some classes; however, these classes have a low penalty and as can be seen from Fig. 11, the gi-FIFO policy does not adapt to the required percentile SLA as well as pairwise interchange causing the total penalty incurred in gi-FIFO to be higher.

Fig. 5: Typical comparison of FIFO and simulated annealing.

Fig. 6: Total penalty incurred in FIFO and Online schedules with varying input rates.

Fig. 7: Penalty assigned to each incoming request for two classes with different conformance requirements.

Fig. 8: Adaptation of algorithm to favor requests with higher penalty, each run with varying input request rate.
as shown in Fig. 10. Also shown in Fig. 11 is that Tabu search with pairwise interchange performs as well as simulated annealing with pairwise interchange gaining on the latter for some classes. This is expected as in Tabu search we can expect to obtain a number of schedules more than that in simulated annealing owing to the tabu list where a configurable past number of schedules are stored and each new schedule is checked against that list and if there is any match, the schedule is discarded and another schedule is obtained in its place. We also compare the penalties obtained with simulated annealing and tabu search both employing pairwise interchange with varying input rates in Fig. 12. As shown, Tabu search and simulated annealing result in low penalties for almost all of the input rates, with Tabu performing a shade better in most cases. However in a few cases the penalty incurred in Tabu search is slightly less than that in simulated annealing and we attribute this to our implementation of dynamic list sizes, where, if requests queued are very less in number, we do not perform comparison with the tabu list in tabu search and so, the variation is due to the randomness in the input rates and schedule selection in the two simulation runs.

With simulations we have found that pairwise interchange with Tabu search or simulated annealing performs the best among all other variants. The pairwise interchange algorithm finds the optimum schedules over a set number of iterations. So our next question is “How do the heuristic-based variants of the algorithm perform with varying iterations?”. To answer this, we evaluated pairwise interchange based Tabu search with a set number of iterations for two different request rates, results in Fig. 13. As expected, as the number of iterations increase, the total penalty decreases, also the minimum iterations required for the total penalty to be zero is higher for higher input rate.

V. Conclusion and future work

In this paper, we studied the problem of request scheduling in a cloud computing system with geographically distributed data centers hosting multiple applications; the system operates under a global, percentile response time SLA. The SLA calls
for economic penalties if percentile targets are not met. We proposed a novel, distributed request scheduling scheme that aims to minimize the total penalty charged on the cloud. We implemented and evaluated two variants of a heuristic algorithm, one based on simulated annealing and another on gi-FIFO scheduling. Our evaluation has shown that our methods far outperform the commonly deployed FIFO scheduling.

Our future work involves expanding the scope of the problem to include (a) on-demand routing of the requests to appropriate resources, (b) dynamic resource management of the servers in a distributed cloud for both the single-step and multi-step percentile SLA and (c) minimization of response time and power consumption-based penalties with multi-tier applications in a distributed cloud computing system.

REFERENCES