Benchmarking the Problem of Optimal Autonomous Systems Aggregation on Different Computer Architectures

Leszek Borzemski, Michał Danielak, and Grzegorz Kotowski
Institute of Informatics
Wroclaw University of Technology
Wrocław, Poland
e-mail: {leszek.borzemski, michal.danielak, grzegorz.kotowski}@pwr.edu.pl

Abstract*—This paper presents and formalizes the problem of optimal Autonomous Systems aggregation in computer network, and shows how this problem can be calculated in a real-life case on two computer architectures: RISC and CISC. On the one hand, the optimal autonomous systems aggregation problem was formulated as an instance of the minimum set cover NP-hard computational problem. On the other hand, the multiplicity of such calculations to be made using massive, distributed and collected in real-time datasets results in challenges of Big Data analysis. Nowadays, RISC based processors have cornered the market of mobile computing solutions, whereas CICSs are dominating in desktop and server computing. But, a new trend in server design, based on RISC system-on-chip processors is deliberated. Therefore we need to consider both processors especially in current computational problems to choice which computer architecture should be used to run the computations. This paper defines the problem of Autonomous Systems aggregation as a set cover problem, gives a brief overview on CISC and RISC architectures, and presents our performance and topology measurements of the Internet as well as our comparison of computational efficiency of both processor architectures with respect to the size of a computation task. We find that the optimal solution of the AS aggregation problem is extremely demanding and time-consuming, especially for the huge number of ASs. Based on the obtained results, we can state that CISC architecture should be chosen to solve the AS aggregation problem.

Keywords-Autonomous Systems; autonomous systems aggregation; CISC; RISC, performance evaluation; computer network monitoring; Big Data; high performance computing.

I. INTRODUCTION AND MOTIVATION

Nowadays, the Internet has become the main source of any information. One of the key advantages of such information is its almost instantaneous accessibility. Considering the growth of web page complexity level and new technology development, one can observe the constant growth of file sizes that are hosted by web servers (e.g., CD/DVD ISO images, FullHD movies, etc.). As a derivative of size, the obtaining time may of course vary depending on the file. However, it may also vary, considering the same file, depending on the source (this hypothesis is discussed in more detail in Section 5), and the differences in downloading time between sources may extend from minutes to hours. Thus, if the selected web object is mirrored (copied to another location or multiple locations), selection of the suitable source may result in substantial obtaining time reduction. In most cases, web-hosting companies provide a user with a list of their servers located around the world for the manual selection. Such a list usually contains some additional information about the servers such as geographical position or server owner (some companies or organizations mirror their servers mutually), which may help the end-user to manually select the most suitable source. Unfortunately, the choice based on geographical distance or user individual preference is not always the optimal one.

The optimal selection ought to rely on a network structure and its analysis. However, the structure of the Internet is extremely large and immensely complex. Therefore, to simplify the management of such a vast network, it should be divided into smaller parts. One smaller part is called Autonomous System (AS) and it can be described as a set of IP addresses under common administration that has a strictly defined routing schema applied. Each of the Autonomous Systems has its registered and globally unique number (ASN).

A few Interior Gateway Protocols (IGP) are used to transfer data within an autonomous system, while Exterior Gateway Protocols (EGP) such as Border Gateway Protocol (BGP) are generally used to exchange routing information between autonomous systems [1]. What is more, BGP is currently the widely used inter-domain BGP on the Internet.

Exchanging information about network reachability with other systems is the primary function of ASs that use BGP. This information is exchanged between ASs and is kept in BGP tables. Finally, data from BGP tables are sufficient to create a graph that represents connections between autonomous systems [2].

Almost 85 percent of ASs registered around the world (approximately 60 000) are active and providing about 200 000 network subnets. Moreover, the global interconnection number equals almost 120 000 [3].

To make the image clearer, one can make real life analogy and compare the above structure to the example of political geography (see Table I).

At the first glance, the simplest method for analysis of the Internet would be to acquire real-time full characteristics of every single host in every AS. Unfortunately, taking into account only the aforementioned numbers, one can notice that the real time analysis or even the simple monitoring of so complex a structure, even from one point of view, tends to be virtually impossible. What is more, the computational power and network limits (which have been recently both continuously raising) are usually insufficient, too expensive and/or limited. Therefore, to deal with the whole AS network, its structure has to be somehow simplified so that its complexity would significantly decrease. This can be achieved by conducting two simple steps: the selection of ASs representatives and the AS aggregation. The first step is to choose a group of hosts belonging to a single AS as its representatives, while the other consists in reducing the set of monitored ASs by aggregating (grouping) them into entities called MetaASs. What is more, both steps have to be conducted, because choosing even a few representatives for an AS makes a number of few hundred thousands of hosts to observe in total, so the aggregation process is indispensable.

TABLE I. INTERNET AND WORLD TERMINOLOGY COMPARISON

Network	Real life
Internet	World
Autonomous System	Country
Autonomous System Number	Country name
AS routing map	Country road map
AS interconnections	Border checkpoints between countries
AS subnets	Country administrative division

Moreover, not only does applying such a reduction slightly decrease the global number of monitored ASs and consequently the number of representative hosts, but it also gives a compromising Internet structure; simultaneously, so simplified approach is not deprived of drawbacks – the structure constructed by this process strongly depends on the location of an observation source and corresponds only to a certain point of view, so it cannot be simply applied to another part of the structure.

One might wonder whether the aggregation of so vast number of ASs is justified, especially for parallel aggregation. Research indicate that the average number of ASs that constitute an AS-path, i.e., a sequence of intermediate ASs between the sender and the receiver, is generally lower than twenty. In case of BGP prepending, however, this path is deliberately lengthened and the maximum AS path can reach even 160 ASs. What is more, in case of parallel aggregation which is solved in this paper, use of ASs aggregations seems to be even more justifiable, because all upstream (or downstream) ASs of a given AS must be aggregated. And generally, this number of ASs is quite large: for AS7029, for instance, the number of its downstream servers is 150 [3]. This considerable number of ASs to aggregate results in extremely demanding computational problem. This is why this particular problem solved here an instance of a Big Data analysis, which needs a specific approach to run computations in hybrid computer systems consisting of various Instruction Systems Architectures (ISAs) processing components, to provide required computations effectively; these ISAs can be either Reduced Instruction Set Computing [3] or Complex Instruction Set Computing [3].

The rest of the paper is organized as follows. Section II formulates the problem of optimal autonomous systems aggregation; Section III gives an overview on Reduced Instruction Set Computing and Complex Instruction Set Computing processing computer architectures; Section IV presents the experiments performed in the Internet and discusses the results of computations made to find optimal Autonomous Systems aggregations for four hardware and software configurations and different numbers of ASs on Internet paths. The paper concludes with a brief review of related work in Section V, and a conclusion summarizing our work and presenting future research plans in Section VI.

II. AUTONOMOUS SYSTEMS STRUCTURE REDUCTION

The reduction of the whole AS structure needs to be performed in a few steps.

A. *Removal of Transitive Autonomous Systems*

The static information about the Autonomous Systems (provided, for instance, by Bates et al. [4]) contains the types that AS can be applied to, i.e., *ORIGIN* (having content), *TRANSIT* (inter-AS communication only) and *ORG+TRN* (hybrid). As we are interested only in web content, the transit part of the network is unnecessary to monitor and may be treated as transparent.

Unfortunately the “transit-only part” does not exceed a few percent of the whole network, so the reduction process has to be more radical.

B. *AS Characteristics*

The vast number of parameters may be used to characterize a single network node. The most common are response times of the remote host, such as Round Trip Time (RTT), Transfer Control Protocol (TCP) connect time, or HyperText Transfer Protocol (HTTP) response time. These parameters may vary inside AS; so, AS characteristics is usually based on their average values.

Round trip time is based either on Internet Control Message Protocol (ICMP) or User Datagram Protocol (UDP) and it is usually the simplest way of determining the response time of a remote host. Unfortunately, not every host replies to *Echo request* (mostly due to administrative policies) which is essential to obtain the RTT. In problems of Web systems, slightly more useful are *TCP connect time* and *HTTP response time*, because in contrary to RTT they are always available; namely, every web server accepts connections incoming to adequate TCP port. TCP connect time describes the time that is needed for establishing the connection (by using the three-way handshake), whereas the HTTP response time is the time between sending the first bytes of the request, e.g., *GET* (see Section 9.3 in [5]) and receiving the first bytes of the response, e.g., *200 OK* (see Section 10.2.1 in [5]). Yet, the three aforementioned examples of characteristics are rather basic, and one may use more complex (comprising of the basic ones) characteristics.

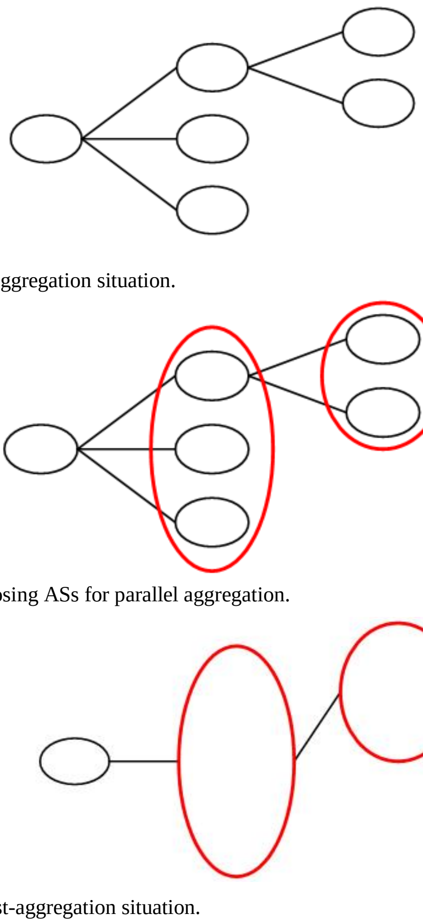
C. *Aggregation*

The subsequent step consists in achieving a simplified structure of ASs that would allow to monitor and/or analyze ASs in (almost) real-time; this process is called the aggregation. The similarity of two AS paths can be described as the longest path segments these paths share. For example, if the destinations are located 9 AS hops away from the source and the paths towards them are the same through 8 AS hops, it can be said that the paths are very similar. In the terms of the real-life example: if we travel from Warsaw to London or Paris, we have to use mostly the same way through Germany and Belgium. As time and distance are roughly the same we may say that our journey is "the same".

Considering the aforementioned example, two cases of aggregation can be distinguished:

1. Siblings (parallel) grouping – grouping ASs directly upstream adjoined to the same AS (see Figs 1a÷c).
2. Parent-child (serial) grouping – path shortening, grouping ASs that are mutually adjoined (up- and downstream); see Figs. 1 a-c.

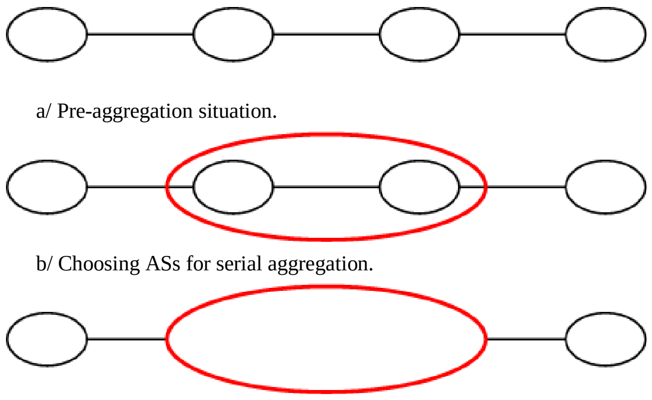
This paper deals only with the first case.


a/ Pre-aggregation situation.

b/ Choosing ASs for parallel aggregation.

c/ Post-aggregation situation.

Figure 1. Siblings grouping.


a/ Pre-aggregation situation.

b/ Choosing ASs for serial aggregation.

c/ Post-aggregation situation.

Figure 2. Serial grouping.

In this work we focus only on the first case – parallel grouping. Let

$$A = \{1, 2, \ldots, N\} \tag{1}$$

Be the set of sibling ASNs and

$$A_i \in 2^A \tag{2}$$

be Meta-AS, i.e., a subset of sibling ASNs fulfilling

$$\forall_{i \in A_k} \left| t(i) - \frac{\sum_{j \in A_k} t(j)}{|A_k|} \right| \le \Delta t \tag{3}$$

where:

$$t : A \to R_+ \tag{4}$$

is the considered performance index (RTT); $t(i)$ is the value of the performance index for i -th AS, whereas Δt is a limiting parameter. In other words, we assume that performance indices for ASs forming a MetaAS cannot deviate from the average by more than Δt. This assures that the aggregated ASs have similar values of performance indices – in the worst case, the difference between the extreme values of these parameters in MetaAS equals to $2\Delta t$.

One can formulate the optimal aggregation problem as follows:

Given (1), (4), find the optimal set of MetaASs C^*:

$$C^* = \arg\min |C| \tag{5}$$

where

$$C \subseteq 2^A \tag{6}$$

such that

$$\bigcup_{A_k \in C} A_k = A \tag{7}$$

$$\underset{A_i \in C, A_j \in C, A_i \neq A_j}{\forall} A_i \cap A_j = \varnothing \tag{8}$$

$$\underset{A_k \in C}{\forall} \underset{i \in A_k}{\forall} \left| t(i) - \tau(k) \right| \leq \Delta t \tag{9}$$

where:

$$\tau(k) = \frac{\sum_{j \in A_k} t(j)}{\left| A_k \right|} \tag{10}$$

is the performance index of a Meta-AS. Constraint (7) states that the sum of all MetaASs should give the set of all ASs, while constraint (8) assures that each AS can only belong to one MetaAS. Finally, constraint (8) states that all MetaASs should fulfill (3).

One can notice that the AS aggregation problem (5) – (10) is an instance of the minimum set cover problem, which is NP–hard. The suboptimal solution algorithms of the generic minimum set cover problem are studied in literature, e.g., in [6][7], but they are not applicable as we search the best solution. Therefore, we solve problem (5) – (10) by means of a brute-force (exhaustive search) algorithm that finds the best solution and offers an easy design but is a long-running time approach, unfortunately.

III. PROCESSING COMPUTER STRUCTURES

Reduced Instruction Set Computing (RISC) and Complex Instruction Set Computing (CISC) are two major computer architectures that can be successfully used to solve the optimal aggregation problem. Despite numerous initial differences between them, these two architectures have become somewhat dependent on each other and the proven solutions from first have gradually started to be adapted in the other and vice versa.

In comparisons of RISC and CISC architectures we can compare them either in qualitative or quantitative way [3]; the former examines the issues of high language support and uses the very large scale of integration (by combining thousands of transistors into single chips) in order to create an integrated circuit, while the later compares applications' size and their execution speeds. This work is a quantitative comparison whose main task is to suggest architecture for the most efficient data processing in the problem of optimal aggregation.

Every quantitative comparison of RISC and CISC architectures should take into account two problems: there is no pair of RISC and CISC machines that would be directly comparable in terms of levels of technology, complexity of compilers or cost and it is impossible to define a set of test programs that would unambiguously evaluate the performance of examined machines [8]. Nevertheless, because the aim of this research is to discover the most efficient architecture for the optimal aggregation problem, the latter problem in this paper seems to be inessential.

The subsequent part of this paper compares the initial assumptions made in CISC and RISC architectures and provides a brief outline of two operating systems that have been used on both architectures to find the optimal hardware and software combination for the optimal aggregation problem.

A. *Main Differencies*

CISCs are defined as the computers with a full set of computer instructions designed to provide needed capabilities in the most efficient way. It was later discovered, however, that by reducing the aforementioned full sets of instructions to those most frequently used, computers would be able to get more tasks done in a shorter amount of time. This reduced approach was defined as RISC. Table II presents the short comparison of those two architectures.

TABLE II. A BRIEF COMPARISON OF RISC AND CISC ARCHITECTURES IN TERMS OF THEIR INITIAL ASSUMPTIONS; SOURCE [9].

CISC	RISC
Primary goal is to complete a task in as few lines of assembly as possible	Primary goal is to speedup individual instruction
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" Incorporated in instructions	Register to register: "LOAD" and "STORE" are Independent instructions
Small code size	Large code sizes
Variable length Instructions	Equal length instructions which make pipelining possible

B. *IBM AIX and Red Hat Enterprise Linux on RISC and CISC Architectures*

Both studied processing structures support Linux operating system which should also be considered in the choice of computing platform. What is more, one should compare all the features and characteristics of evaluated operating systems in order to choose 'the best' operating system. Naturally, this task is virtually impossible to tackle. Nevertheless, this section tries to briefly characterize a few differences between two systems that have been used in this research: Red Hat Enterprise Linux (RHEL), a commercial Linux distribution developed by Red Hat, Inc., and IBM Advanced Interactive eXecutive (AIX).

The first major difference between evaluated operating systems is their availability; namely, RHEL have been ubiquitous and widely used by various business entities in the recent years, while AIX have been used only by these who have chosen IBM's POWER-based IT solutions. Not only can RHEL run on computers with IA-32, IA-64, POWER processors, but it can also be used as an operating system for many embedded devices, mobile computers and servers. On the other hand, IBM has done their bests so that AIX could be more available; currently, one can find AIX-based computers on a large suite of systems, ranging from blades, through small rack units and up to full-rack systems.

Scalability is another significant factor. According to Galvin, RHEL is capable to simultaneously use only 64 cores and 256 GB if Random Access Memory (RAM) (128 cores and 256 GB) on CISC (RISC) architecture. In comparison, AIX can use four times more cores and over thirty times more memory (256 cores and 8192 GB memory). What is more, comparing the actual performance of operating systems is an extremely challenging task, because benchmarks are generally designed to measure either hardware or application performance.

Operating systems that have been used for solving the AS aggregation task could be characterized in more detail in terms of features, such as computer virtualization, code debugging, as well as system installation and administration. Nonetheless, given the characteristics of the AS aggregation task, there is no need to discussed all of them; detailed comparison of the functionality of AIX and RHEL can be found in [9][10].

C. Hardware Platforms

All experiments were conducted in the Department of Distributed Computer Systems at Wroclaw University of Technology. The tests were conducted on three servers equipped with RISC processors, successively referred to as: RISC-AIX, RISC-RHEL, RISC-RHEL-ND and one CISC-based processor IBM Blade CISC server referred to as CISC-RHEL.

TABLE III. HARDWARE AND SOFTWARE CONFIGURATIONS

Component	RISC-AIX	RISC-RHEL	RISC-RHEL-ND	CISC-RHEL
Processor	4.0 GHz POWER6	4.0 GHz POWER6	4.0 GHz POWER6	3.0 GHz Xeon 4C
# of CPUs	2	2	2	2
CPU cores	2	2	2	4
CPU type	x64	x64	x64	x64
Memory	4GB	4GB	4GB	8GB
HDD storage	73.4 GB SAS 10 000 rpm	73.4 GB SAS 10 000 rpm	n/a	73.4 GB SAS 10 000 rpm

ND suffix stands for no disc; this means that every piece of equipment that is marked with ND does not contain internal storage. Moreover, in case of RISC, 4 cores CPUs are in fact 2 double-cored CPUs. The detailed hardware description of the above-mentioned servers is shown in Table III.

All RISC-based computers (those with RISC prefix) contain two dual-core POWER6 processors (see Fig. 3). Each POWER6 processor consists of two 4.0 GHz cores that are capable of two-way Simultaneous MultiThreading (SMT), which allow multiple independent threads to be executed, and one AltiVec (previously called IBM VMX) which is a floating point and integer Single Instruction Multiple Data (SIMD) instruction set accelerator. Therefore, in this paper, we are trying to evaluate to what extent the use of AltiVec acceleration may improve the general performance of POWER-based computers in the optimal AS aggregation problem.

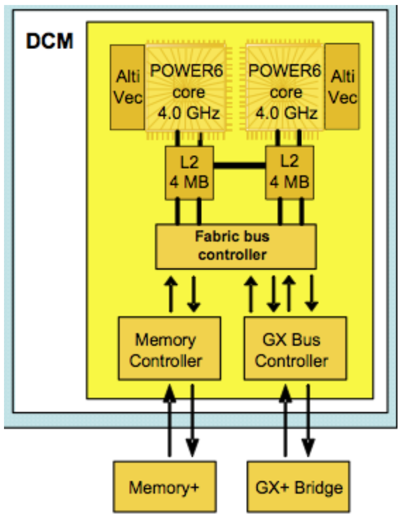


Figure 3. POWER6 [IBM].

The configuration of server RISC-RHEL-ND differs from the one of RISC-RHEL only in terms of lack of internal disk storage and its operating system is run using local network from the external disk array. This configuration enabled us to examine whether the lack of internal disk storage and thus the need to install an operating system on the external disk array, affects (and if it does, to what extent) the efficiency of numerical calculations in the optimal aggregation problem.

IV. BENCHMARKING EXPERIMENTS

This section is divided into two parts: the first part presents a sample method that may be used to monitor a group of ASs; the other describes all the experiments that were conducted in order to select the best combination of software and hardware architecture for solving the optimal AS aggregation problem.

But, before we explain the procedures carried out, let us go back to the very beginning of this paper. In the first section, we have formulated the hypothesis concerning the relationship between resource download time and the server location in the Internet. We have carried out a simple experiment to prove it. To achieve this, we have randomly selected 30 web servers with identical lists of available resources (in this case, we used official mirror servers with Linux Ubuntu 13.04 installation image file; its file size equals to 735MB). In the subsequent step, we have measured download times of the aforementioned resource from the all considered web servers. Fig. 4 presents average download times of the resource. To make the figure more readable, we have used the logarithmic scale.

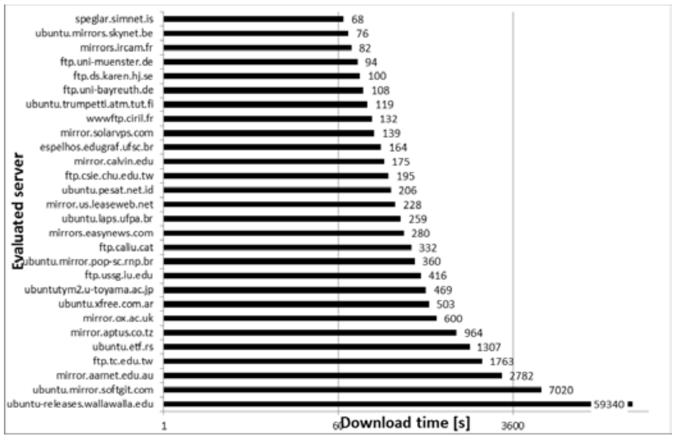


Figure 4. Download times.

The conducted experiment may be also considered as a method for the evaluation of AS' representative's characteristics (and consequently the characteristics of a whole AS). However, one ought to remember that even the simple selection of the download file is extremely crucial and may have a significant influence on the accuracy of the results and thus their usability for further processing. Therefore, the resource should be selected carefully; namely, the size of the resource should be neither too small (because the measurements should be greater than a statistical error) nor too large (the measurements cannot excessively burden the evaluated representative hosts of a considered AS).

The experiments described in Sections IV.A and IV.B were performed using the same input data (i.e., RTT) for evaluated ASs. In the first experiment, the values are converted to milliseconds and stored as integers. In the second experiment, however, the values are converted to seconds (with fractional part) and stored as single precision floating-point numbers.

Unfortunately, due to a substantial time-consuming nature of the experiment for both, integer and floating point numbers, the double precession numbers were not considered in this work. Nevertheless, they are another worth pondering example to be examined in the future work.

A. *AS Aggregation: Integer Case*

Fig. 5 shows the relationship between the aggregation time and the number of ASs for the integer case. One may notice that the aggregation time grows exponentially. What is more, it turned out that CISC computer is able to solve the aggregation task much more efficiently. Take the last measurement (wherein the number of ASs is 25) for instance. It took 169 days (almost half a year) for RISC-AIX server to generate the desired MetaASs, whereas its CISC counterpart solved this problem in barely 128 days (less than four and the half months). One should realize that the difference is immense in terms of execution time and this quick evaluation survey speaks for CISC solutions even for so small a size of the aggregation problem.

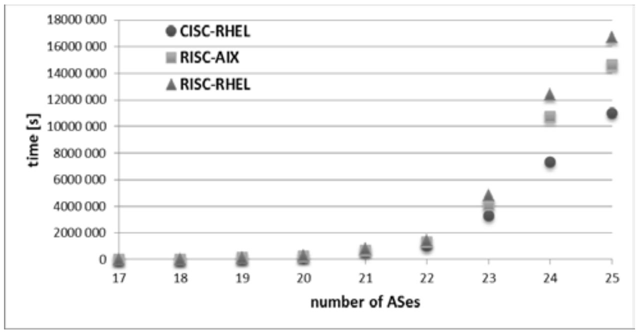


Figure 5. Aggregation time vs. the number of ASs. Integer case, computed on a single core .

The choice of an operating system for calculation is also essential. Namely, calculations conducted on the same model of server are performed slightly faster on AIX than on RHEL.

It turned out that the results obtained for the RISC-RHEL-ND and RISC-RHEL were strikingly similar for all considered cases, so they were neglected and they are not presented in any figure. This also showed that the speed of calculations depends only on the central processing unit.

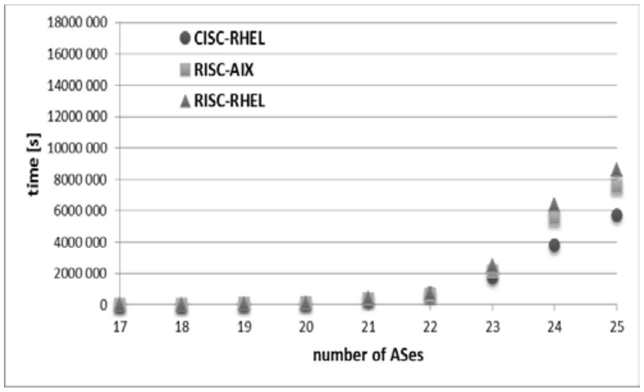


Figure 6. Aggregation time vs. the number of ASs. Integer case, computed on two cores.

Fig. 6 presents the relationship between the aggregation time and the number of ASs subjected to that process, calculated on two cores for the integer case. One may notice a dramatic decrease of execution time that is needed to find the optimal solution. Time needed to obtain the result for 25 ASs by RISC machine is almost two times smaller than that for the aggregation running on a single core. In this case, CISC is also much more efficient than RISC-equipped computers. The execution time for CISC-RHEL and RISC-RHEL for 23 ASs equals 20 and 29, respectively. The selection of the operating system has also the impact on the calculation time; this impact is similar to that of presented in the Fig. 5.

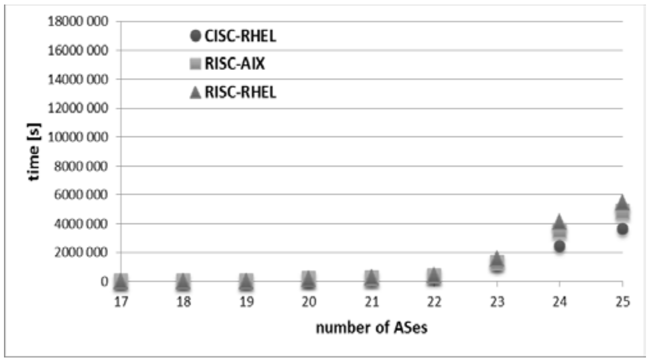


Figure 7. Aggregation time vs. the number of ASs. Integer case, computed on four cores.

Fig. 7 presents the relationship between the aggregation time and the number of ASs subjected to that process, calculated on four cores for the integer scenario. One may notice a decrease (comparable with that of the previous figure) of execution time that is needed to find the optimal solution. To illustrate the chasm in the computational speed between RISC and CISC, one may compare the results obtained for 24 ASs and 25 ASs by RISC-AIX and CISC-RHEL, respectively – they are almost the same.

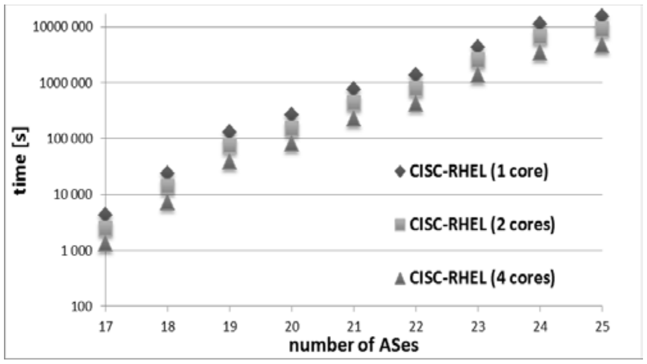


Figure 8. Aggregation time vs. the number of ASs. CISC architecture multicore comparison for the integer case, presented on logarithmic scale.

Fig. 8 shows calculations on a logarithmic scale, conducted for the CISC machine for all considered integer cases. This illustrates two things: the exponential computational complexity of the aggregation problem, and constancy of the increase of power achieved by involving additional CPU cores.

B. AS Aggregation: Floating Point Case

Figs. 9, 10, and 11 present the results obtained for characteristics represented as floating points for a single core, two cores and four cores scenarios, respectively. The situation turned out to be completely different from the research conducted for ASs' characteristics represented as integers. The POWER-based computers turned out to be the more efficient than the evaluated CISC machine.

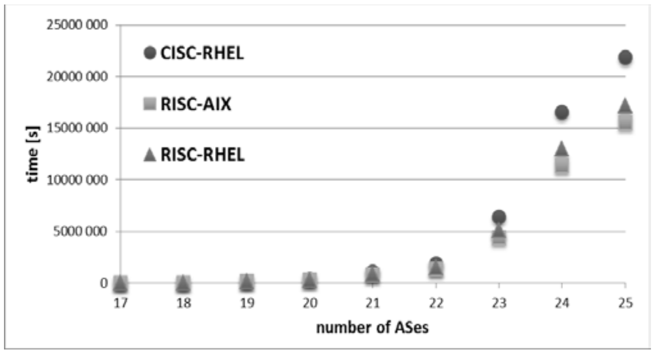


Figure 9. Aggregation time vs. the number of ASs. Floating point case, computed on a single core.

The choice of an operating system for calculation is as essential as in the case of integer scenarios; therefore, all conclusions that can be drawn from the conducted experiments are similar to those from the previous subsection.

The most surprising hypothesis that can be proposed, however, is that with the simultaneous increase in both the number of cores and the number of ASs to be aggregated, the differences between two considered architectures gradually commence to narrow.

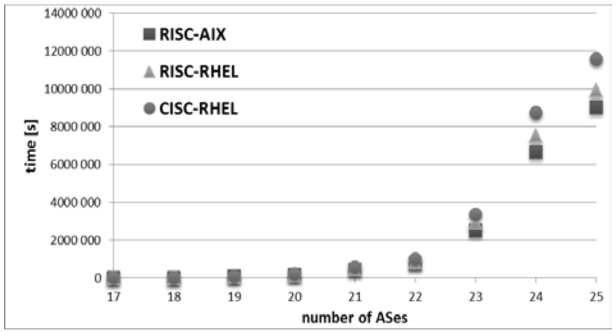


Figure 10. Aggregation time vs. the number of ASs. Floating point case, computed on two cores.

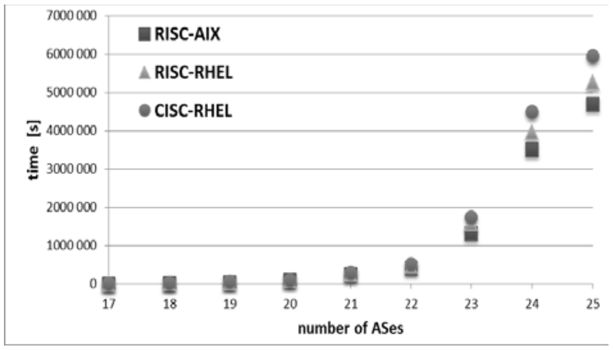


Figure 11. Aggregation time vs. the number of ASs. Floating point case, computed on 4 cores.

C. Matrix Multiplication

Performing numerous arithmetic calculations is a prerequisite to perform the AS aggregation. We benchmark the speed of arithmetic calculation, by performing simple matrices multiplications. The aim of this subsection is to examine, whether the floating-point accelerator that can be found in RISC architecture has (and if it does, to what extent) an impact on the calculation speed.

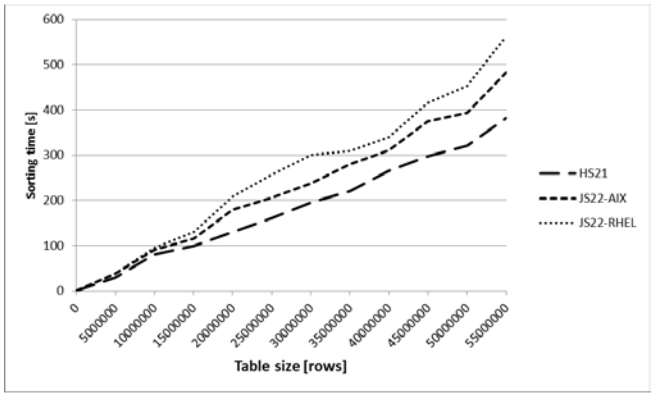


Figure 12. Multiplication time vs. matrix size.

Fig. 12 presents the relationship between the time needed to perform multiplication of two square matrixes and their size (*N*). Both matrixes consisted of *N*×*N* floating point numbers. One should note that choice of the operating was essential. Strictly speaking, calculations conducted on the same server are performed slightly faster on AIX than on RHEL. Furthermore, due to the use of floating point numbers, RISC computers were able to use their accelerators and thus needed less time to obtain the correct results. This proved that the built-in floating-point accelerator might significantly increase the speed of calculations.

V. Related Work

Diversity of techniques and approaches that deal with the problem of AS topology decomposition may be found in the literature. Dimitropoulos et al., for instance, use information retrieval techniques (in this case, this is an expert system along with simple text classification techniques) to analyze data from Internet Routing Registers [11]. Mostly, however, scientists tackle the problem of AS aggregation by analyzing BGP-derived AS graphs that represent connections between ASs. Ge et al. use Consumer-to-Provider (C2P) relationships to classify ASs into seven groups (tiers); basically, their concept is based on the idea that all providers should be in higher ASs than their clients [12]. Subramanian et al. extend this approach by taking into account peer-peer relationships [13].

Nevertheless, these approaches usually suffer from a slight drawback; namely, they rely exclusively on BGP-derived AS graphs. Unfortunately degree-based classification may result in an incomplete AS topology and bring about imprecise decomposition. The approach presented in this paper tries to solve this problem in a slightly different manner; namely, by using BGP-derived AS interconnection structure we try to identify and measure the characteristics of ASs' representatives in order to use them in the aggregation process.

The literature focusing on AS-level Internet topology research is extensive and has been quickly growing in the past few years [14][15][16][17][18]. Roughan et al. [14] and Jyothi et al. [15] discuss the most important problems found in the previous work on AS topology phenomenon and propose some advancements towards improving measuring and modeling in this research. Chen et al. [16] touch problems of revealing AS-topology in the context of Peer-to-Peer (P2P) networks development. Borzemski and Nowak [17] introduce the novel algorithm to predict performance of web servers using revealed information about AS topology between a web server and a web client. Durairajan et al. [18] propose to build a comprehensive and geographically accurate map of the physical Internet, including ASs, for future research and operational applications. This proposal, as well as, several other efforts in Internet measuring, analysis, and modelling are instances of Big Data analysis conducted in Fan et al. [19]. Big Data paradigm promises new scientific levels of Internet research. According to the reference [3] in [19], two main goals of analyzing Big Data are to develop effective methods that can accurately predict the future observations, and at the same time to gain insight into the relationships between the features and response for scientific purposes. Taking into account these goals, Borzemski [20] proposes Web Performance Mining (WPM) - a new dimension in web mining analysis - to predict web server performance as experienced by the web user. Another approach, which is based on geostatistical Turning Bands Methods, has a similar aim and is described in [21]. Borzemski [20], and Borzemski and Kamińska-Chuchmała [21] explore large databases containing continuously collected records of performance measurements of Internet connections. References [22][23] present our distributed measurement infrastructure MWING that allows to measure the Internet from multiple vantage points, like in [13].

VI. Conclusion and future work

Based on the obtained results, it can be stated that CISC architecture should be chosen to solve the AS aggregation problem discussed in this paper. Not only are they capable of producing correct results more rapidly than RISC-based computers, but also they are less expensive. One may, however, argue that ASs' characteristics are usually fraction values; so RISC-based computers could be the solution. Nonetheless, it turns out that in such cases it is somehow possible to convert such data into integer values without losing any information.

The operating system may also influence the general performance for it turns out that IBM AIX can slightly more efficient than Red Hat Enterprise Linux.

The optimal solution of the AS aggregation problem is extremely demanding and time-consuming, especially for the huge number of ASs. Nevertheless, in a real-life case scenario, we do not always have to find the optimal result because the suboptimal is often satisfactory. Therefore, we can simplify the whole aggregation by performing it gradually. This could result in the significant reduction in the time needed to perform the aggregation (the processing time increases exponentially to the size of the problem).

Nonetheless, due to the complexity of the examined phenomena, the research presented in this paper were conducted only for the single precision floating-point number representation. In the future work, we would like to focus on conducting another experiment that compares differences in performance (if there are any) between single and double precision numbers.

Considering the results of the conducted experiments in case of the floating-point scenarios, it can be assumed that the more cores of multicore processor are used in the optimal AS aggregation, the smaller the differences between the obtained performance indices become. Therefore, in the future research, one should consider the comparison of the architecture performance in more complex computing structures such as High Performance Clusters. In this scenario, three cases should be considered:

1. Homogeneous CISC-based Clusters.
2. Homogeneous RISC-based Clusters.
3. Heterogeneous Clusters (consisting of both CISC and RISC based computational nodes).

ACKNOWLEDGMENT

The authors gratefully acknowledge partial support of project EU COST Action IC0805: Open Network for High-Performance Computing on Complex Environments.

REFERENCES

[1] J. Hawkinson and T. Bates, “Guidelines for creation, selection, and registration of an autonomous system”, Request for Comments 1930, Internet Engineering Task Force, 1996.

[2] Y. Rekhter, T. Li, and S. Hares, “A border gateway protocol 4”, Request for Comments 4271, Internet Engineering Task Force, 2006.

[3] W. Stallings, Computer Organization and Architecture: Designing for Performance. Prentice Hall Inc., a Simon Schuster Company, 2009.

[4] T. Bates, P. Smith, and G. Huston, “CIDR Report for 27 May 14, 2014, vol. 2014, http://www.cidr-report.org/as2.0/ [accessed: 2014-05-27].

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext Transfer Protocol” – HTTP/1.1, June 1999, Request for Comments: 2616.

[6] V. Cgvatal, “A greedy heuristic for the set-covering problem”, Mathematics of Operations Research, vol. 4 no. 3, 1979, pp. 233–235.

[7] V. Vazirani, Approximation Algorithms. Springer-Verlag, 2001.

[8] C. Chen, G. Novick, and K. Shimano, “The intellectual excitement of computer science: RISC architecture”, http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/2000-01/risc/ [accessed: 2014-05-24].

[9] P. Galvin, “Pete’s all things Sun: comparing Solaris to Red Hat Enterprise and AIX, 2010.

[10] P. Galvin, “Pete’s all things Sun: comparing Solaris to Red Hat Enterprise and AIX — Virtualization Features”, 2011.

[11] X. Dimitropoulos, G. Riley, and K. Claffy, “Revealing the autonomous systems taxonomy: The Machine Learning Approach”. Proc. of Passive and Active Measurement Conference (PAM), pp. 91-100, 2006.

[12] Z. Ge, D. Figueiredo, S. Jaiwal, and L. Gao, “On the hierarchical structure of the logical Internet graph. Proc. SPIE vol. 4526, pp. 208-222, 2001.

[13] L. Subramanian, S. Agarawal, S. Rexfors, and R. H. Katz, “Characterizing the Internet hierarchy from multiple vantage points”. IEEE INFOCOM, 2002.

[14] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, “10 lessons from 10 years of measuring and modeling the Internet's autonomous systems”. IEEE Journal on Selected Areas in Communications 29(9): 1810-1821, 2011.

[15] S. A. Jyothi, A. Singla, B. Godfrey, and A. Kolla, “Measuring and understanding throughput of network topologies”. CoRR abs/1402.2531, 2014 [accessed: 2014-05-24].

[16] K. Chen, D.R. Choffnes, R. Potharaju, Y. Chen, F.E. Bustamante, D. Pei, and Y. Zhao, “Where the sidewalk ends: extending the Internet AS graph using traceroutes from P2P users”. IEEE Trans. Computers 63(4): 1021-1036, 2014.

[17] L. Borzemski and Z. Nowak, Using autonomous system topological information in a web server performance prediction”. Cybernetics and Systems 39(7): 753-769, 2008.

[18] R. Durairajan, S. Ghosh, X. Tang, P. Barford, and B. Eriksson, “Internet atlas: a geographic database of the internet”. HotPlanet '13 Proceedings of the 5th ACM workshop on HotPlanet, pp. 15-20, 2013.

[19] J. Fan, F. Han, and H. Liu, “Challenges of Big Data analysis”. National Science Review Advance Access publication Feb 6, 00:1-22, 2014.

[20] L. Borzemski, “The use of data mining to predict web performance”. Cybernetics and Systems 37(6): 587-608, 2006.

[21] L. Borzemski and A. Kaminska-Chuchmala, “Distributed web systems performance forecasting using Turning Bands method”. IEEE Trans. Industrial Informatics 9(1): 254-261, 2013.

[22] L. Borzemski, L. Cichocki, and M. Kliber, “Architecture of multiagent Internet measurement system MWING Release 2”. Lecture Notes in Artificial Intelligence, vol. 5559, pp. 410-419, 2009.

[23] L. Borzemski, “The experimental design for data mining to discover web performance issues in a Wide Area Network”. Cybernetics and Systems 41(1): 31-45, 2010.