APIO 2014 Technical Report

General information

On APIO 2014 we had 29 countries competing in two days, and 619 official contestants (though some small amount of contestants didn't show).
There were 3 tasks and about 7500 submissions.

Hardware:

Primary:
25 IBM BladeCenter HS22, each with 2 x Intel Xeon X5570 @ 2.93GHz, 12GB RAM, 2 x 128GB HDD in RAID1

Secondary (Amazon EC2):
1 instance of c3.2xlarge: 8 vCPUs, each is HT from Intel Xeon E5-2670 v2 (Ivy Bridge), 15GB RAM, 2x80GB SSD in RAID1
40 instances of c3.large: 2 vCPUs, each is HT from Intel Xeon E5-2670 v2 (Ivy Bridge), 3.75GB RAM, 2x16GB SSD in RAID1

Software:

Operating system: Ubuntu Server 13.10, 64-bit
C/C++ compiler: GNU GCC 4.8.1
FreePascal compiler: FPC 2.6.2-5
Contest management software: CMS (http://cms-dev.github.io/)

The primary setup

Primary main server. Contained all services from CMS except workers. All services presented in one instance except for CWS. There where 5 CWS instances. Balancing between CWS instances was done via Nginx with "sticky" module (http://gravitronic.com/compiling-the-nginx-sticky-session-module-in-ubuntu/, https://code.google.com/p/nginx-sticky-module/). We couldn't use ip_hash directive, because the server was behind NAT and all incoming requests were coming from the Internet gateway, so from the server's point of view, all contestants were on the same ip address.

PostgreSQL was tuned to allow more connections (we ended at 300) and hot standby replication mode was configured (http://wiki.postgresql.org/wiki/Binary_Replication_Tutorial). To allow more connections default Ubuntu value for SHMMAX had to be increased:

sysctl -w kernel.shmmax=134217728 (or add it to /etc/sysctl.conf)

Secondary main server. Contained the same setup, but everything was turned off with exception for PostgreSQL instance, acting as a replication slave for main server.

Both servers had different external IP-addresses. Switching from the main server to the secondary main server would take several minutes and contestants would have to use new URLs.

Worker machines. We were worried about evaluating solutions in time, so we tried to get as

much workers as we can. As we had 8-core machines with a lot of memory, we had an thought, that running several solutions in parallel on such machine wouldn't be too harmful. And in fact, it wasn't harmful, tests showed this. So we decided to run 4 instances of workers on each machine. Because of this during the first day of contests, we had 23 * 4 = 92 workers. But by the middle of the first day, when the rush hour passed, we understood that this number could be several times lower and even in that case we wouldn't have a queue - we overestimated the number of submissions and time needed for evaluation. That's why we decided to stop additional worker services and return to one-worker-per-machine configuration on the next day. And to make the competition fair, we have rejudged all submissions of day one on the new configuration during the night between contest days, and only result of one submission has changed. During day two we had no problems at all - just had to remember to answer contestants' questions.

The secondary setup (on Amazon EC2)

It is similar to the primary one, but because workers had only 2 CPU cores, there was only one worker service per machine from the very beginning. The system was up and was used to handle some unofficial contestants. In case of troubles on the main system we could switch all contestants here in several minutes.

Tuning CMS

During preparation we did several changes in CMS, some were dirty hacks, some where accepted as pull requests, and on some of them we will work to make them acceptable into the master branch.

The most important change is skipping test cases in subtasks after the first failed test case and including a subtask into another one (i.e. skipping larger subtasks if the smaller one has failed). More information here: https://github.com/cms-dev/cms/issues/258. We used this during the contest, and I think this significantly improved the evaluation time, and because of this we had never seen any queue during the contest. I plan to work on this more.

Another one that really helped in preparing the contest, is the loader for Polygon contests format. See here: https://github.com/cms-dev/cms/issues/258. It still needs to be improved, and I hope after some time it will also be merged.

Several dirty hacks also can be mentioned: automatically applying token for all submissions (https://github.com/artikz/cms/commit/5673c735207af4fd6831aaeaf7a8023154fc916f) and improving performance of contest overview page in AWS for handling large number of submissions (https://github.com/artikz/cms/commit/2c6b72a1cbcc70a98ad27d8fa487acc7d0a7029d, hope I will be able to make it in a more beautiful way).

There were also some fixes and some less important but useful features. Some of them are already in the master branch, some of them are waiting for their turn, and some of them are not reusable in current form.

Handling virtual contests

APIO is conducted in form of virtual contests. Every country has it's own contest on the same set of tasks in the desired time. CMS has builtin feature for handling virtual contests, but it required some hacks to be suitable for us. The native CMS feature allows to start a contest

and let a contestant start his timeframe by himself.

So, if we would wanted to use this feature directly, we had two options:
1. start one contest with all contestants included. It's very simple and convenient, but in this case all contestants would be able to login at any time during two days, which is not desirable.
2. start 29 virtual contests, limiting their timeframes to fit the schedule. Oh, I'm too old for this...

So we wanted the simplicity of case 1 and manageability of case 2. And we decided to use case 1 and hack the CMS in the following way:

1. disallow contestants to start timeframe by themselves (remove "Start" button and don't forget to comment out the StartHandler in CWS);
2. create a small service for country supervisors, which allowed them to set starting_time by themselves. It was a webpage, accepting supervisor's email and password (different for all supervisors), and if they were correct, it sets the starting_time for contestants of this supervisor to the moment at which the button was pressed.

You can see these hacks here:
https://github.com/artikz/cms/commit/e3d8a55201d2b0b6b0ed72ba98855449d63e410e

Conclusion

Overall, the contest went very smoothly: there were no unexpected misbehaviour of the system (big thanks to CMS developers!), and we overestimated the load and the time for evaluation, so we had a huge resources reserve (big thanks to KBTU!).