Project Final Presentation :

Settings Manager & Internal Trigger

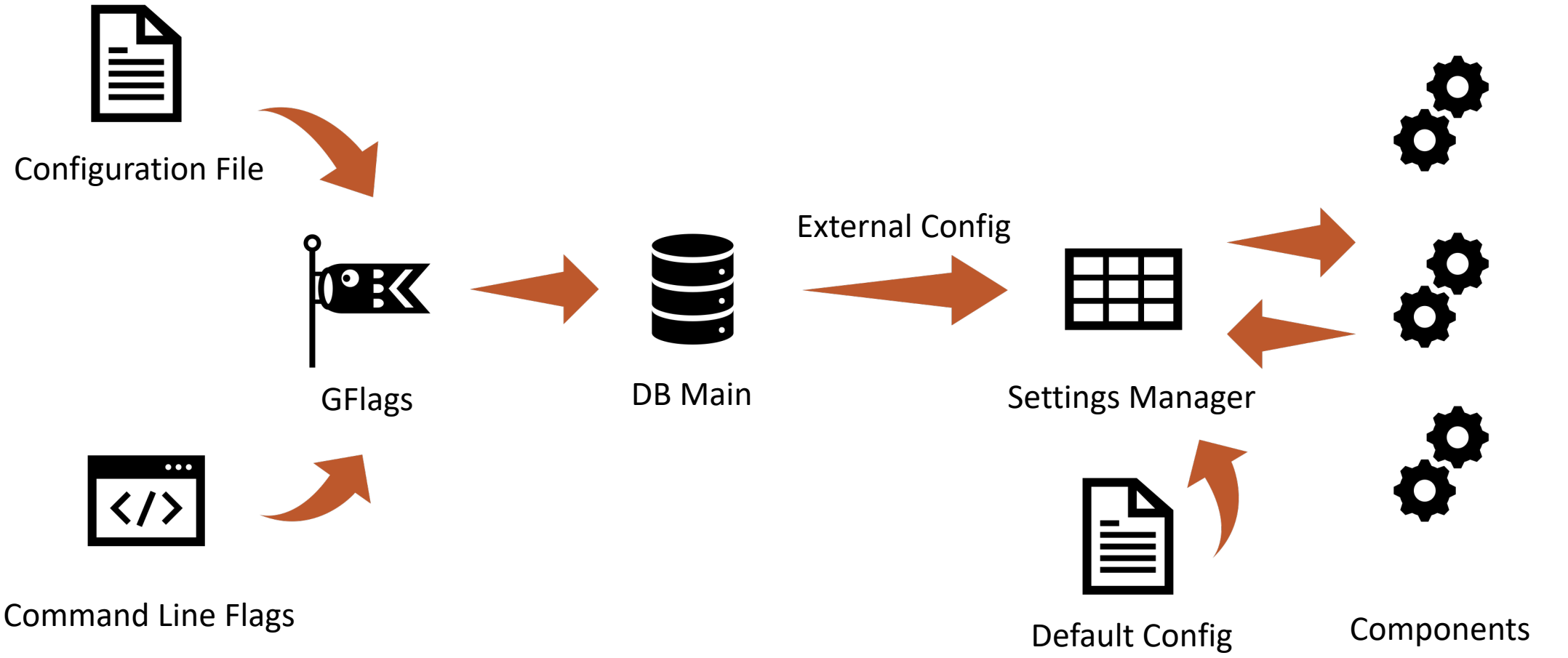
Weichen Ke, Wenhao Huang, Yuze Liao

2019.05.06

Project Goals

- **125% (Done)**:
  - Implemented initialization of configuration knobs, standardized APIs, basic callback triggers, etc.
  - Implemented several static setting knobs, e.g. port number; Implement basic compile-time check.
  - Implemented several tunable setting knobs, e.g. the buffer pool size limit. Support identifying whether a knob is tunable or not.
  - Implemented action context that stores the information of a tuning action. Finished the callback framework.
  - Implemented the first version of DB main object and bootstrapping logic.

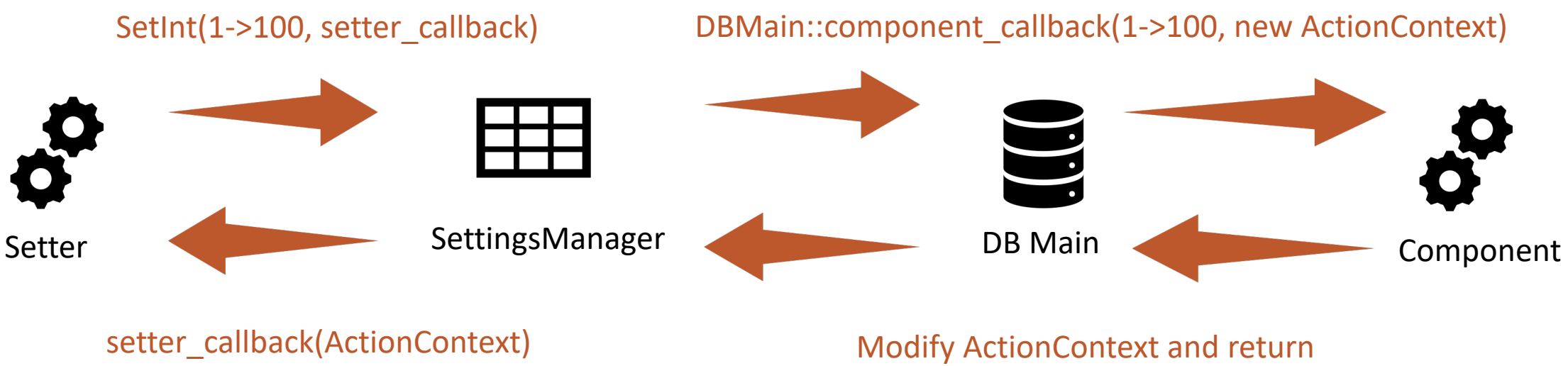
System Bootstrap



System Bootstrap

- Now it is easy to add custom parameters in tests.
- Instead of “`--my_new_param=something`” , you can make a map and pass it to DB Main/SettingsManager.
- `map[Param::my_new_param] = ParamInfo(“something”);`
- This makes testing easier.

Callback Flow



Macros

- Settings only need to be defined once by users with macros.
  - Macros are expanded in different places
    - Gflags
    - Parameter enumeration class
    - SettingsManager
- **Easy to use, Easy to read.**

```
SETTING_int(
  port,
  'Terrier port (default: 15721)',
  15721,
  1024,
  65535,
  false,
  MainDatabase::EmptyCallback
)
```

Documented Macro Expansions

- Macro expansions in the Settings Manager is magical.
- So we tell the reader what is going on when we use magic.

```
/*
 * Populate gflag values to param map.
 * This will expand to a list of code like:
 * param_map_.emplace(
 *     terrier::settings::Param::port,
 *     terrier::settings::ParamInfo(port, terrier::type::TransientValueFactory::GetInteger(FLAGS_port),
 *                                  "Terrier port (default: 15721)",
 *                                  terrier::type::TransientValueFactory::GetInteger(15721), is_mutable));
 */

#define __SETTING_POPULATE__            // NOLINT
#include "settings/settings_common.h"   // NOLINT
#include "settings/settings_defs.h"     // NOLINT
#undef __SETTING_POPULATE__             // NOLINT
```

Testing and Evaluation

- **Unit Test**:
  - Check the consistency of settings manager and pg_settings table
  - Check if the callbacks are correctly called when changing a parameter
  - Check if the callbacks set ActionContexts correctly
  - Check type integrity and constraints enforcement (immutable parameters, min/max bounds…)

Future work

- **Get rid of the physical pg_settings table**
  - There is no such table in Postgresql…
- **More strict compile time check**
  - Whether the parameter is mutable
  - Whether the user called correct setter (int/decimal/string…)
  - May need more macro expansions.

Thank you!