配置选项

guc.c:750
类型定义：src/backend/utils/misc/guc_tales.h
可用选项列举：src/backend/utils/misc/guc.c

读取配置流程：
Postmaster.c:PostmasterMain()
  guc.c:InitializeGUCOptions()
    guc.c:build_guc_variables() // 为配置选项构造排序(name)数组
    guc.c:InitializeOneGUCOption() // 使用默认值初始化
    gui.c:InitializeGUCOptionFromEnvironment()
  Parse conmand-line options
    SetConfigOption()
  guc.c:SelectConfigFiles()
    guc-file.c:ProcessConfigFile()
      guc-file.c:ProcessConfigFileInternal()
        guc-file.c:ParseConfigFile()
          guc-file.c:ParseConfigFp()
            guc-file.c:ParseConfigDirectory()
            guc-file.c:ParseConfigFile()

guc.c:GetConfigOption()
guc.c:SetConfigOption()

syslog_split_messages: git show fc201dfd95059fd2fef9862a2fd09cfab42c9bf7
添加配置选项需要修改的文件：
doc/src/sgml/config.sgml 添加配置选项说明 doc/src/sgml 目录下相关文件
src/backend/utils/misc/guc.c 添加配置选项
src/backend/utils/misc/postgresql.conf.sample 添加配置选项
以及相关的使用

如果不是修改 PostgreSQL 源文件，而是添加 Extension，则可参考
citus:src/backend/distributed/shared_library_init.c:_PG_init()
调用过程：
DefineCustomXXXVariable()
  init_custom_variable() // 分配 config 结构体并填充 generic 字段
  define_custom_variable() // 将新变量插入到 GUC 变量数组
    add_guc_variable()
注意：使用 DefineCustomXXXVariable() 定义的配置参数如果要写到 postgresql.conf 文件，则名称中必须有分隔符 '.'，否则启动 PG 的时候会报错 unrecognized configuration parameter(如果将 DefineCustomXXXVariable()放到 postmaster.c:824 SelectConfigFiles()前面就可以，但很少将自定义配置参数放到如此前面，这样如果没有 . 分隔符，调用

SelectConfigFiles() 就会出错)。具体参考 src/backend/utils/misc/guc-file.c:2061 行。例如 custom.max_login_users，而不是 max_login_users。

查询执行流程

```
postgres.c:PostgresMain()
  exec_simple_query()
    pg_parse_query()
      raw_parser()
    pg_analyze_and_rewrite()
      parse_analyze()
        transformTopLevelStms()
      pg_rewrite_query()
        QueryRewrite()
    pg_plan_queries() {
      pg_plan_query()
        planner()
          // (*planner_hook) 调用自定义的 planner
          standard_planner()
            subquery_planner()
              create_plan()
    }// pg_plan_queries
    CreatePortal()
    PortalDefineQuery()
    PortalStart() {
      PortalGetHeapMemory()
      ChoosePortalStrategy()
      1)    case PORTAL_ONE_SELECT:
        CreateQueryDesc()
        ExecutorStart()
          // (*ExecutorStart_hook) 调用自定义的 executor
          standard_ExecutorStart()
            InitPlan()
      2)    case PORTAL_ONE_RETURNING || PORTAL_ONE_MOD_WITH
      3)    case PORTAL_UTIL_SELECT
      4)    case PORTAL_MULTI_QUERY
    }//PortalStart
    PortalSetResultFormat()
    CreateDestReceiver()
    SetRemoteDestReceiverParams()
    PortalRun() {
      PortalGetHeapMemory()
      1)  case PORTAL_ONE_SELECT || PORTAL_ONE_RETURNING ||
```

```
                PORTAL_ONE_MOD_WITH || PORTAL_UTIL_SELECT
            PortalRunSelect()
                ExecutorRun()
                    // (*ExecutorRun_hook)
                    standard_ExecutorRun()
                        ExecutePlan()
                            ExecProcNode()
                                case xxx: ExecXXX()
        2)  case PORTAL_MULTI_QUERY
            PortalRunMulti()
                // process a plannable query
                ProcessQuery()
                    ExecutorStart()
                    ExecutorRun()
                    ExecutorFinish()
                    ExecutorEnd()
                // or process utility functions
                    PortalRunUtility()
                        ProcessUtility()
                            //(*ProcessUtility_hook)
                            standard_ProcessUtility()
                                case XXX: PerformXXX()
    }// PortalRun
    PortalDrop()
  }// exec_simple_query
}//PostgresMain
```

Citus 使用 PostgreSQL 共享内存

```
[root@nobida147 citus]# grep Shmem -r src/|grep -v Binary
src/backend/distributed/master/worker_node_manager.c:static Size WorkerNodeShmemSize(void);
src/backend/distributed/master/worker_node_manager.c:static void WorkerNodeShmemAndWorkerListInit(void);
src/backend/distributed/master/worker_node_manager.c:   RequestAddinShmemSpace(WorkerNodeShmemSize());
src/backend/distributed/master/worker_node_manager.c:   shmem_startup_hook = WorkerNodeShmemAndWorkerListInit;
src/backend/distributed/master/worker_node_manager.c:WorkerNodeShmemSize(void)
src/backend/distributed/master/worker_node_manager.c:WorkerNodeShmemAndWorkerListInit(void)
src/backend/distributed/master/worker_node_manager.c:   WorkerNodesHash = ShmemInitHash("Worker Node Hash",
src/backend/distributed/worker/task_tracker.c:static Size TaskTrackerShmemSize(void);
src/backend/distributed/worker/task_tracker.c:static void TaskTrackerShmemInit(void);
src/backend/distributed/worker/task_tracker.c:  RequestAddinShmemSpace(TaskTrackerShmemSize());
src/backend/distributed/worker/task_tracker.c:  shmem_startup_hook = TaskTrackerShmemInit;
src/backend/distributed/worker/task_tracker.c:TaskTrackerShmemSize(void)
src/backend/distributed/worker/task_tracker.c:TaskTrackerShmemInit(void)
src/backend/distributed/worker/task_tracker.c:  LWLockAcquire(AddinShmemInitLock, LW_EXCLUSIVE);
src/backend/distributed/worker/task_tracker.c:          (WorkerTasksSharedStateData *) ShmemInitStruct("Worker Task Control",
src/backend/distributed/worker/task_tracker.c:          ShmemInitHash("Worker Task Hash",
src/backend/distributed/worker/task_tracker.c:  LWLockRelease(AddinShmemInitLock);
[root@nobida147 citus]#
```

仔细阅读

citus/src/backend/distributed/master/worker_node_manager.c // 使用 PG 共享内存

postgresql/src/backend/storage/ipc/shmem.c // PG 共享内存实现

postgresql/src/backend/utils/hash/dynahash.c // PG 动态哈希实现

postgresql/src/backend/utils/hash/hashfn.c // PG 哈希函数

主要是：ShmemInitHash() 和 ShmemInitStruct()

Citus 使用 PostgreSQL MemoryContext

citus/src/backend/distributed/worker/task_tracker.c:TaskTrackerMain():140