Simuler l'optique adaptative sur GPUs

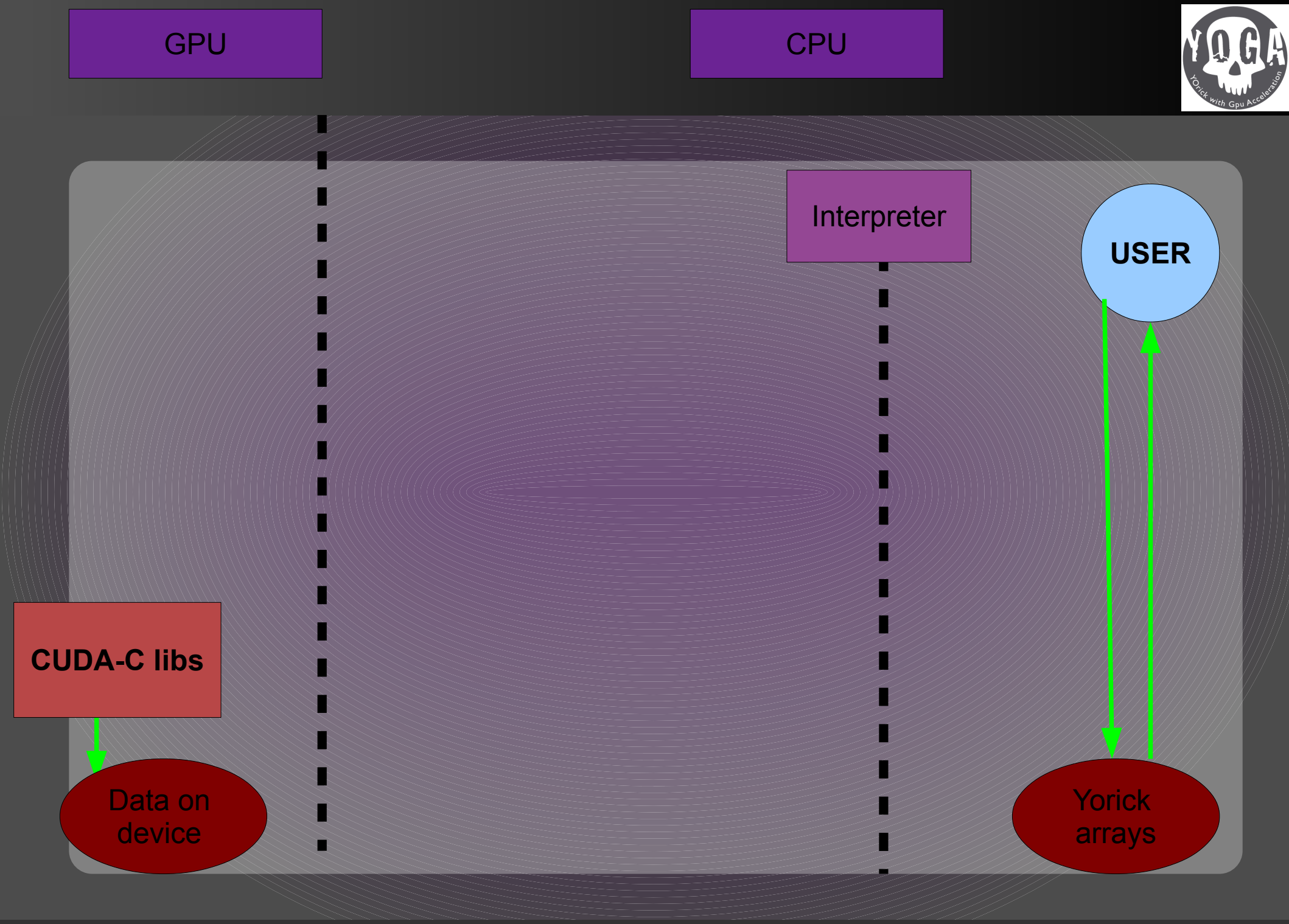
Damien Gratadour

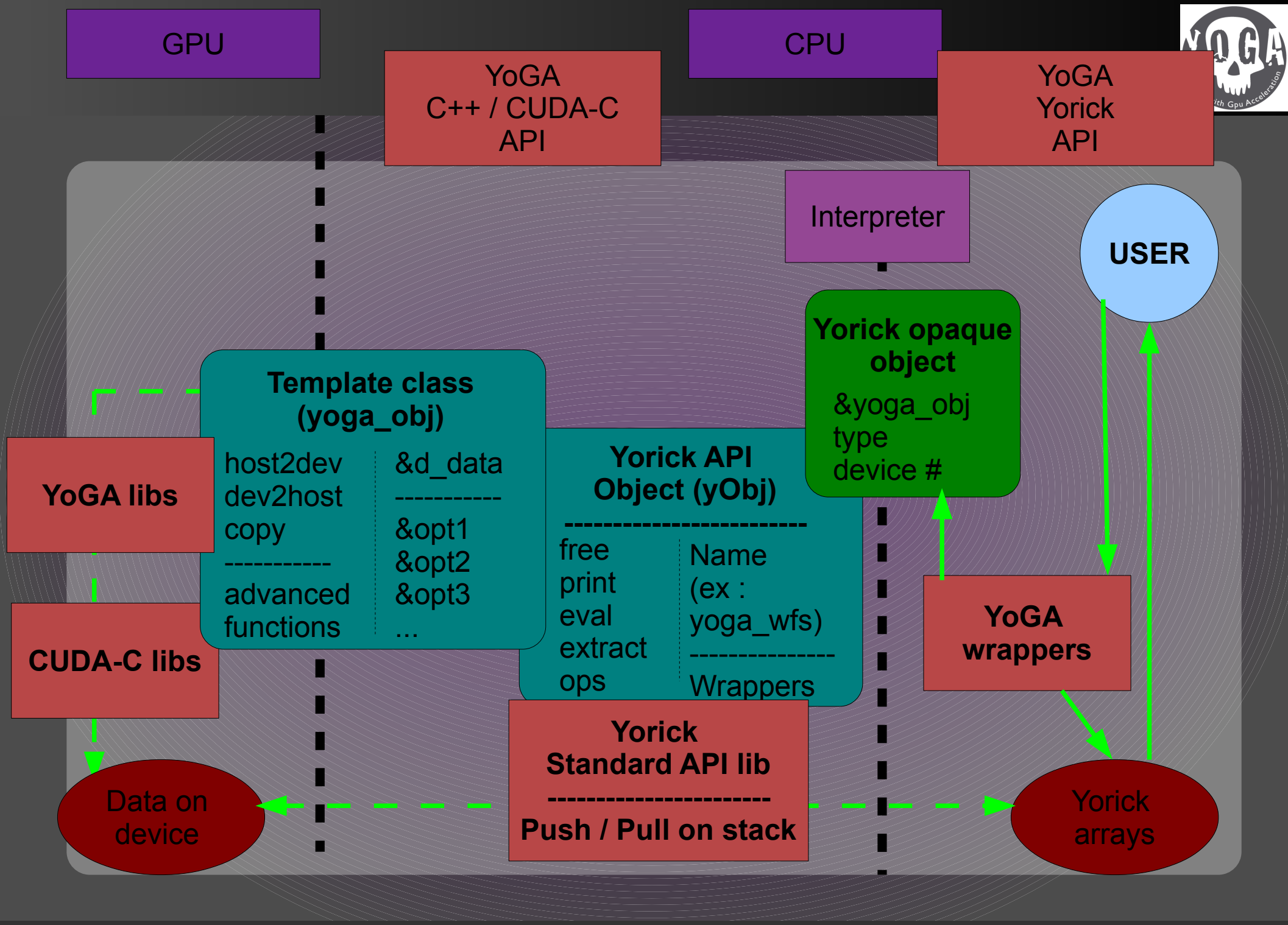
Plan

- **L'extension YoGA_AO**
  - Structures de données et algorithmes
- **Fonctionnalités & performance**
- **Développements futurs**
- **Le projet COMPASS**
- **Démo !**

YoGA_AO

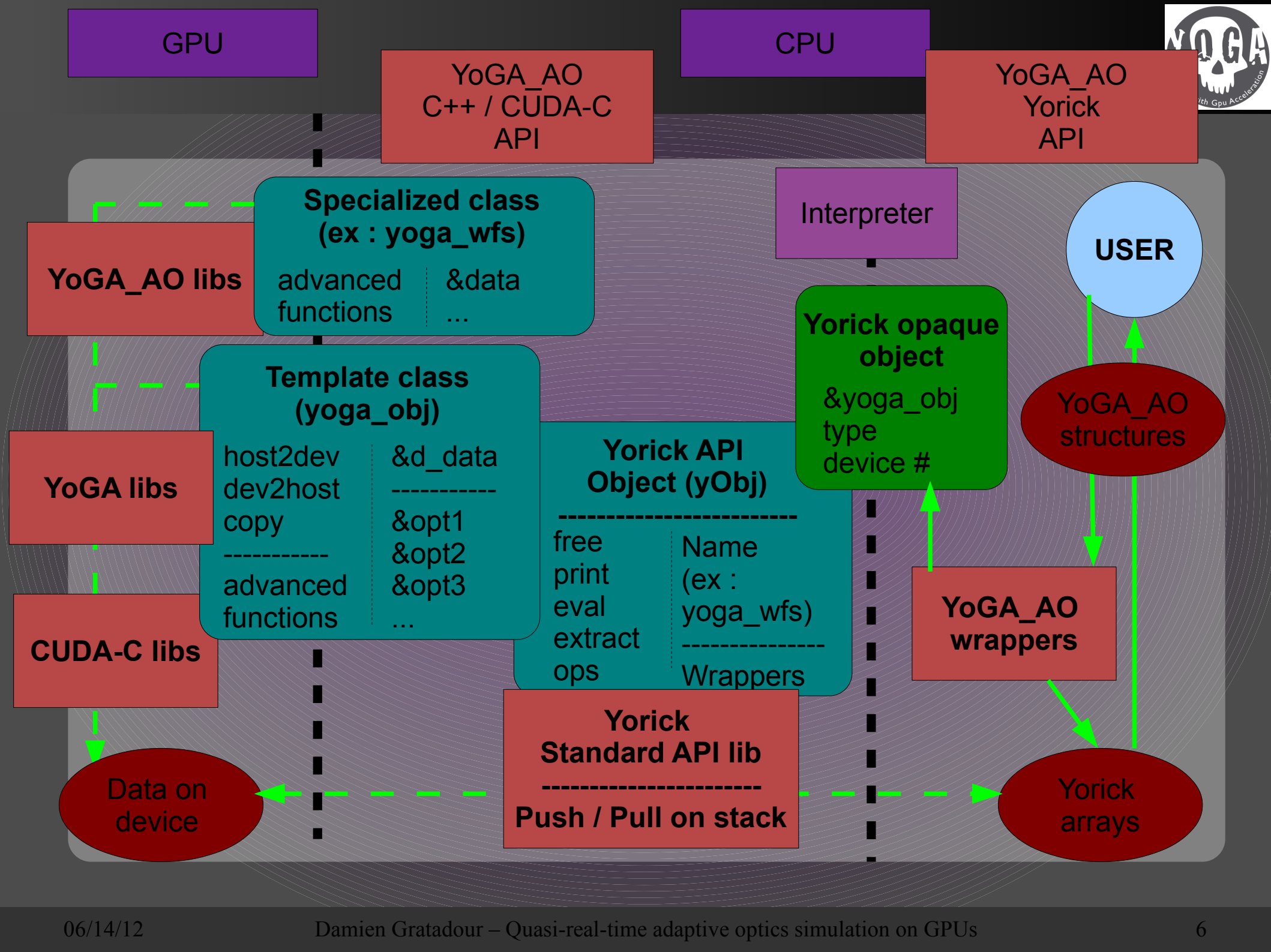
- **Extension de YoGA pour l'OA**
  - Basée sur l'utilisation de la classe yoga_object (shared library) pour les fonctionnalités de base
  - Classes custom pour la simulation de l'atmosphère, les optiques, ASO de type SH, sources guides, etc …
  - Une autre librairie partagée en Cuda-C + son extension Yorick
  - Accès simplifié à tous les paramètres depuis une session Yorick (utile pour le debug / diagnostic + affichage)
- **Mode script ou GUI**
  - *Templates* de scripts optimisés pour la performance
  - GUI utilisant le *binding* yorick-python + GTK
- **Principales fonctionnalités liées à l'OA**
  - Multiple couches de turbulence
  - Analyseur type Shack-Hartmann (NGS + LGS)
  - Calcul des pentes avec divers algorithmes
  - Algorithmes de contrôle simpliste type “moindres carrés”
  - Modèle réaliste de miroir type piezzo (couplage inter-actionneur, etc ...)


GPU
CPU
Interpreter
USER
CUDA-C libs
Data on device
Yorick arrays


GPU
CPU
YoGA
C++ / CUDA-C
API
YoGA
Yorick
API
Interpreter
USER
Yorick opaque
object
&yoga_obj
type
device #
Template class
(yoga_obj)
host2dev
dev2host
copy
---
advanced
functions
&d_data

&opt1
&opt2
&opt3
...
Yorick API
Object (yObj)
free
print
eval
extract
ops
Name
(ex :
yoga_wfs)

Wrappers
YoGA libs
CUDA-C libs
YoGA
wrappers
Yorick
Standard API lib
Push / Pull on stack
Data on
device
Yorick
arrays


GPU
CPU
YoGA_AO
C++ / CUDA-C
API
YoGA_AO
Yorick
API
Specialized class
(ex : yoga_wfs)
advanced
functions
&data
...
YoGA_AO libs
Interpreter
USER
Yorick opaque
object
&yoga_obj
type
device #
YoGA_AO
structures
Template class
(yoga_obj)
host2dev
dev2host
copy
advanced
functions
&d_data
&opt1
&opt2
&opt3
...
YoGA libs
Yorick API
Object (yObj)
free
print
eval
extract
ops
Name
(ex :
yoga_wfs)
Wrappers
YoGA_AO
wrappers
CUDA-C libs
Yorick
Standard API lib
Push / Pull on stack
Data on
device
Yorick
arrays

YoGA_AO

- **Exemple de la génération de turbulence**
- **Basé sur l'utilisation des *yoga objects* et des fonctionnalités de la STL telles que *maps* ou *vectors***

```
class yoga_tscreen {
 public:

  int                     device;       // The device #
  yoga_phase              *d_tscreen;   // The phase screen
  yoga_obj<float>         *d_tscreen_o; // Additional space of the same size as the phase screen
  yoga_obj<float>         *d_A;         // A matrix for extrusion
  yoga_obj<float>         *d_B;         // B matrix for extrusion
  yoga_obj<unsigned int> *d_istencilx; // stencil for column extrusion
  yoga_obj<unsigned int> *d_istencily; // stencil for row extrusion
  yoga_obj<float>         *d_z;         // tmp array for extrusion process
  yoga_obj<float>         *d_noise;     // tmp array containing random numbers
  yoga_obj<float>         *d_ytmp;      // contains the extrude update (row or column)
  long                    screen_size;  // size of phase screens
  float                   amplitude;    // amplitude for extrusion (r0^-5/6)
  float                   altitude;
  float                   windspeed;
  float                   winddir;
  float                   deltax;       // number of row to extrude per iteration
  float                   deltay;       // number of row to extrude per iteration
  //internal
  float                   accumx;
  float                   accumy;
  cudaChannelFormatDesc   channelDesc;  // Channel descriptor for texture memory access

  yoga_obj<cuFloatComplex>  *d_tscreen_c; // Additional space for von karman screen generation
  float                   norm_vk;
  bool                    vk_on;
  yoga_context *current_context;

 public:
  yoga_tscreen(yoga_context *context, long size, long size2, float amplitude, float altitude,
               float windspeed, float winddir, float deltax, float deltay,int device);
  yoga_tscreen(const yoga_tscreen& tscreen);
  ~yoga_tscreen();

  int init_screen(float *h_A, float *h_B, unsigned int *h_istencilx,unsigned int *h_istencily,
                  int seed);
  int extrude(int dir);
  int init_vk(int seed, int pupd);
  int generate_vk(float l0,int nalias);
};
```

```
class yoga_atmos {
 public:
  int                     nscreens;
  map<float,yoga_tscreen *> d_screens;
  float                   r0;
  yoga_obj<float>         *d_pupil;
  yoga_context *current_context;

 public:
  yoga_atmos(yoga_context *context, int nscreens,float *r0,long *size, long *size2,
             float *altitude, float *windspeed, float *winddir, float *deltax, float *deltay,
             float *pupil, int device);
  ~yoga_atmos();

  int init_screen(float alt,float *h_A, float *h_B, unsigned int *h_istencilx
                  ,unsigned int *h_istencily, int seed);
  int move();
};
```

YoGA_AO

- **Autre exemple : yoga_rtc**
- **Depend de 2 classes : *centroiders* et *controlers***
- **Allocation dynamique possible par l'utilisation de *vectors***
- **Coeur "temps-réel" de la simulation**

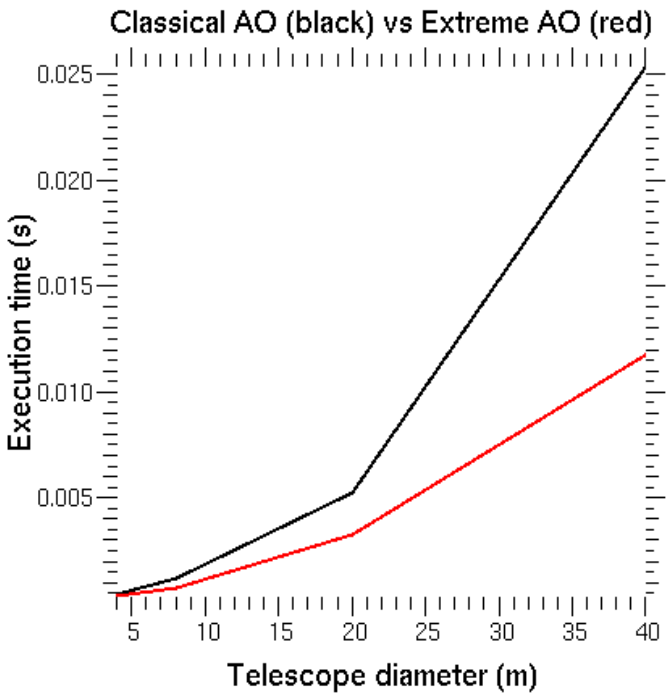
```
class yoga_rtc {
 public:
  int                          device;
  int                          delay;

  vector<yoga_centroider *>  d_centro;
  vector<yoga_controler *>   d_control;

  yoga_context *current_context;

 public:
  yoga_rtc(yoga_context *context);
  yoga_rtc(const yoga_rtc& yrtc);
  ~yoga_rtc();

  int add_centroider(long nwfs, long nvalid,float offset, float scale, long device, char *typec);
  int rm_centroider();

  int add_controler(long nactu, long delay, long device,const char *typec);
  int rm_controler();

  int do_imat(int ncntrl, yoga_sensors *sensors, yoga_dms *ydm);
  int do_imat_geom(int ncntrl, yoga_sensors *sensors, yoga_dms *ydm,int type);

  int do_centroids(yoga_sensors *sensors);
  int do_centroids(int ncntrl, yoga_sensors *sensors);

  int do_control(int ncntrl, yoga_dms *ydm);

};
```

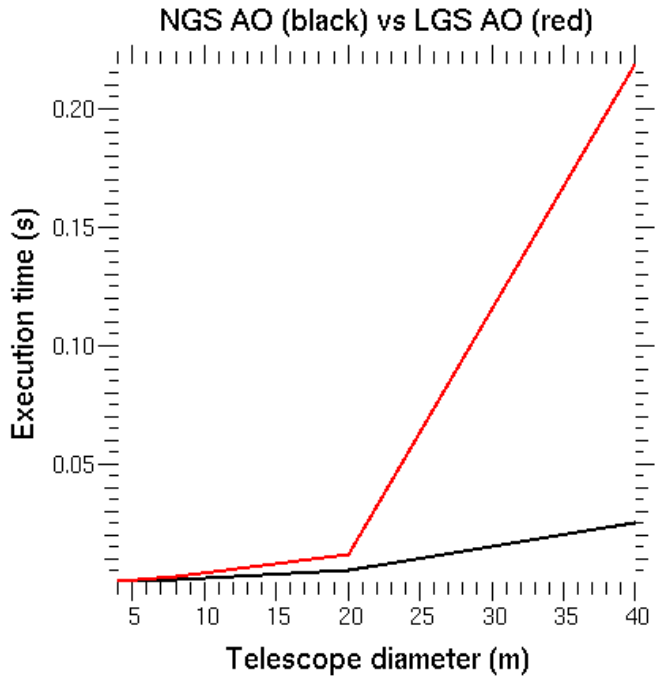
YoGA_AO

- **Fonctionnalités**
  - Génération de la turbulence “au vol” sur des couches multiples à différentes altitudes avec des forces, des vitesses de vent et dess direction différentes
  - Tracé de rayon optimisé dans une direction donnée
  - Possibilité d'inclure des cibles multiples
  - Modèle optimisé d'ASO Shack-Hartmann
  - Modèle réaliste d'étoile laser (à partir de profils mesurés)
  - Différents algorithmes de centre de gravité (COG, seuillé, pondéré, corrélation)
  - Multiple ASO dans de multiple directions (LGS ou NGS)
  - Modèle réaliste de miroir piezzo
  - Approche simple, type “moindres carrés” pour la commande
  - Interface facile d'accès à travers Yorick

Performance (ASO)

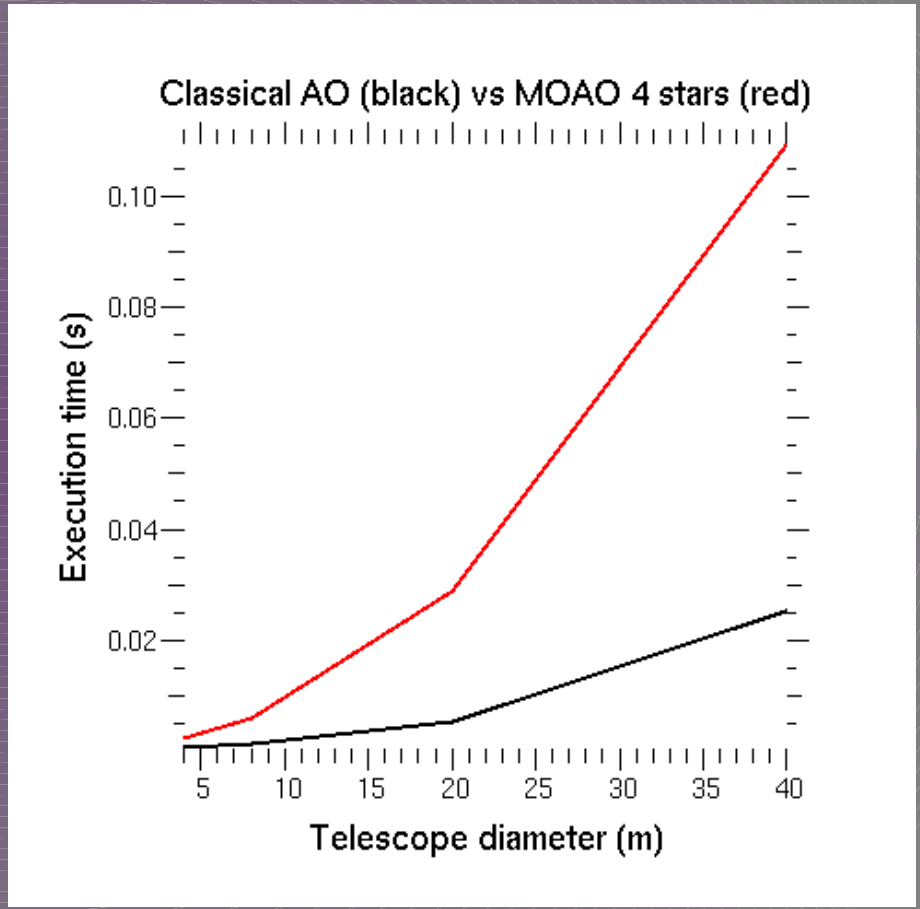
- **Tests sur une carte "*gamer*" : GTX 480**
- **Différents dimensionnements**
  - OA classique (SCAO) : sous-pup Ø ~ 50cm => # sous-pup ~ 2 x diam. télescope
  - OA extrême (XAO) : sous-pup Ø ~ 20cm => # sous-pup ~ 5 x diam. télescope
  - Meilleures performances pour la XAO !
  - XAO : sous-pup plus petites => moins de points de phase par sous-pup => meilleures propriétés vis-à-vis de la mémoire partagée du GPU => gain significatif en performance
- **Cas inverse : OA LGS**
  - LGS: champ des sous-pup plus grand et augmente avec le diam. du télescope (élongation) => dégradation significative des performances





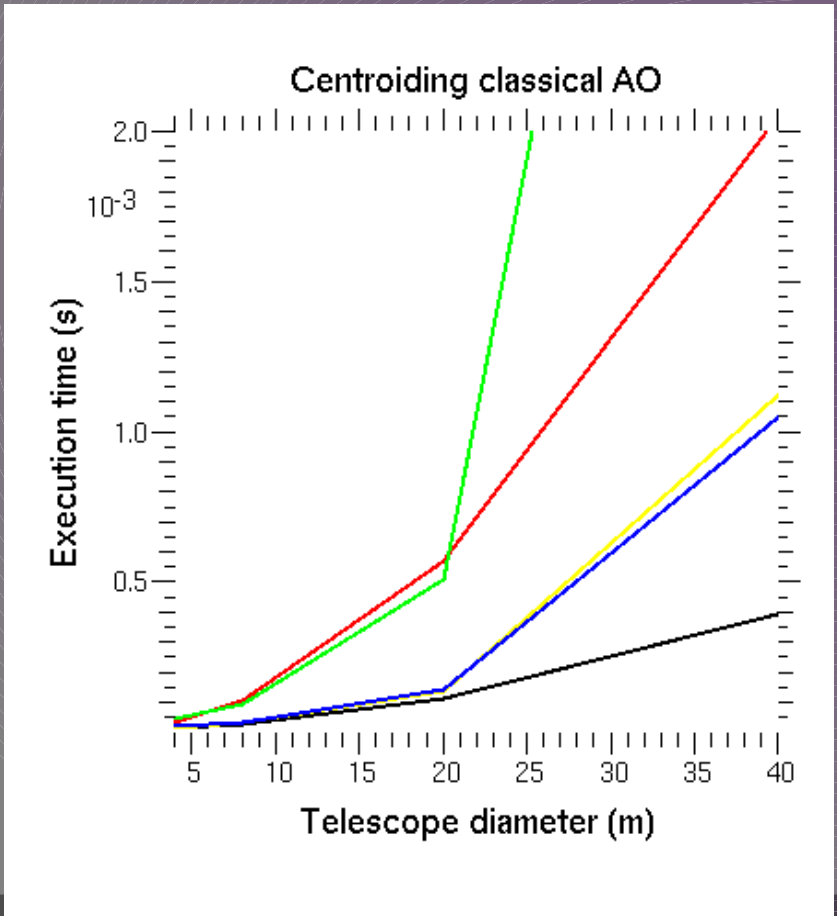
Performance (ASO)

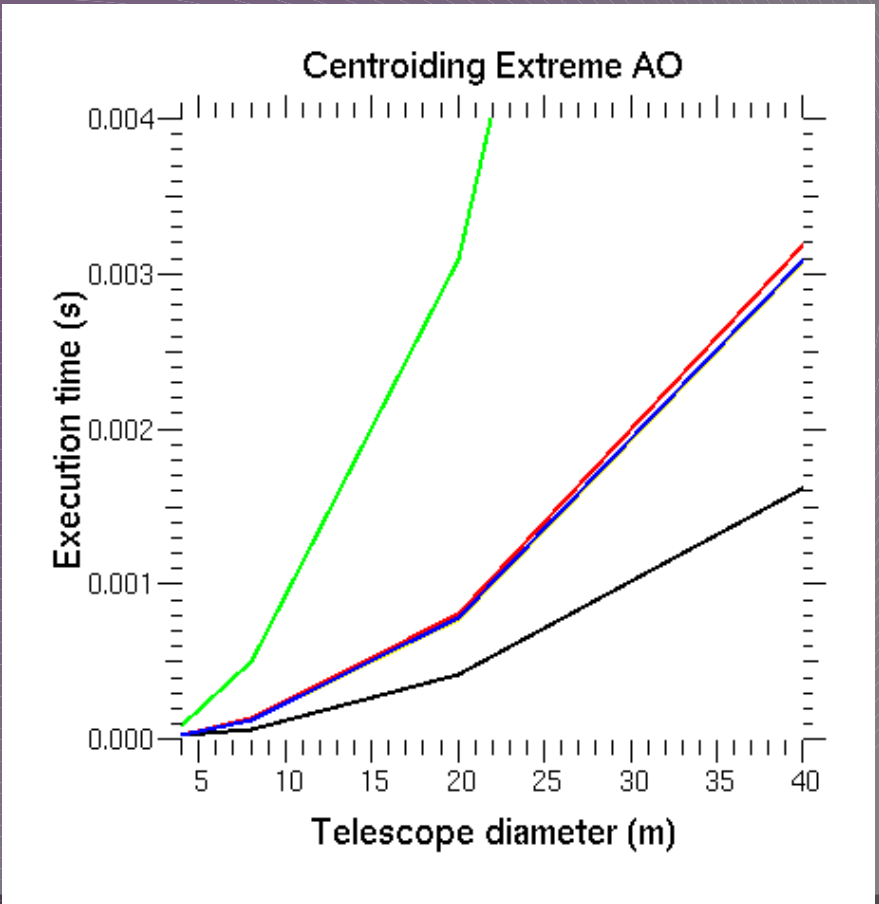
- **Cas de multiple ASO**
  - Pour l'instant, code séquentiel pour le cas à ASO multiples donc aucun gain en performance par rapport à a SCAO
  - Travail en cours sur une meilleur intégration du multi-GPU afin de paralléliser le calcul des images / pentes des ASO sur différents GPU (inclus un entrelaçage calcul / copie fin pour conserver la performance)



Performance (ASO)

- **Centre de gravité**
  - Presque aussi rapide pour la XAO et la SCAO : le code bénéficie à plein des performances du GPU





Performances

Profils d'éxécution d'une boucle SCAO complète (ms)

Tests sur une carte "science" : Tesla C2070

	Turbu	Tracé turbu	Tracé DM	ASO	COG	Contrôle	Tracé cible	Image cible
4m	0.294	0.0144	0.083	0.369	0.021	0.068	0.0151	0.080
8m	0.288	0.0145	0.145	1.66	0.021	0.068	0.0154	1.43
20m	0.289	0.0158	0.330	11.8	0.023	0.070	0.0151	3.26
30m	0.290	0.0158	0.488	26.3	0.023	0.071	0.0163	4.81
40m	0.290	0.0159	0.638	47.8	0.023	0.071	0.0155	6.43

Performances

Même exercice avec une Tesla m2090

	Turbu	Tracé turbu	Tracé DM	ASO	COG	Contrôle	Tracé cible	Image cible
4m	0.284	0.0194	0.118	0.300	0.031	0.071	0.0198	0.114
8m	0.287	0.0197	0.202	1.34	0.032	0.072	0.0204	0.199
20m	0.235	0.0179	0.379	9.36	0.028	0.063	0.0194	0.377
30m	0.252	0.0176	0.550	21.1	0.027	0.065	0.0177	0.546
40m	0.252	0.0179	0.721	38.2	0.027	0.066	0.0168	0.726

Conclusion et perspectives

- **YoGA_AO : une simulation d'OA optimisée pour les GPUs**
  - Environ 1000 iterations/s sont obtenues pour des systèmes de XAO sur des télescopes de 8m : comparable au contrôle temps-réel
  - Environ 100 iterations/s sont obtenues pour des systèmes de SCAO & XAO à l'échelle de E-ELT : sufisamment réaliste pour lancer des campagnes de simulation à grande échelle
  - Code bénéficie à plain de l'architecture des GPU dans certaines config.
  - Interface facile d'accès pour le debug / diagnostic
- **Manque certaines composantes**
  - ASO type pyramide (travail en cours)
  - Algorithmes de contrôle : doivent être pensés et optimisés pour le temps-réel
- **Développements futurs**
  - Intégration complète du multi-GPU (travail en cours) pour étudier les concepts avancés de l'OA (multi-ASO LGS)
  - Outils idéal pour tester des RTC en conditions “nominales” : besoin de définir une interface standard vers les RTC actuellement disponibles.

Demo time !

Projet COMPASS

- **Environnement de développement unifié pour les plateformes hétérogènes**
  - Fédérer les efforts menés au sein de PHASE (LESIA / GEPI + ONERA + LAM + IPAG) pour développer et maintenir une plateforme de développement pour l'OA
  - Partenaire associé : Maison de la Simulation un institut regroupant l'expertise du CEA + CNRS + INRIA + 2 Universités parisienne autour des techniques liées au calcul intensif
  - Collaboration pluridisciplinaire : systèmes d'OA + astrophysique + HPC
  - Produit final : une plateforme haute performance basée sur une intégration complète du software et du hardware dédiée aux architectures hétérogènes.
- **Buts**
  - Une plateforme de développement software : valider des composants clés / tester des nouveaux concepts
  - Un environnement de calcul efficace : faire tourner des simulations à grande échelle
  - Un environnement unifié et optimisé pour PHASE
  - Développer des composants “real-time” (COG / contrôle) pour des architectures massivement parallèles
  - Ouvrir la voie au contrôle de l'OA par des architectures basées sur l'utilisation d'accélérateurs comme les GPU

Projet COMPASS

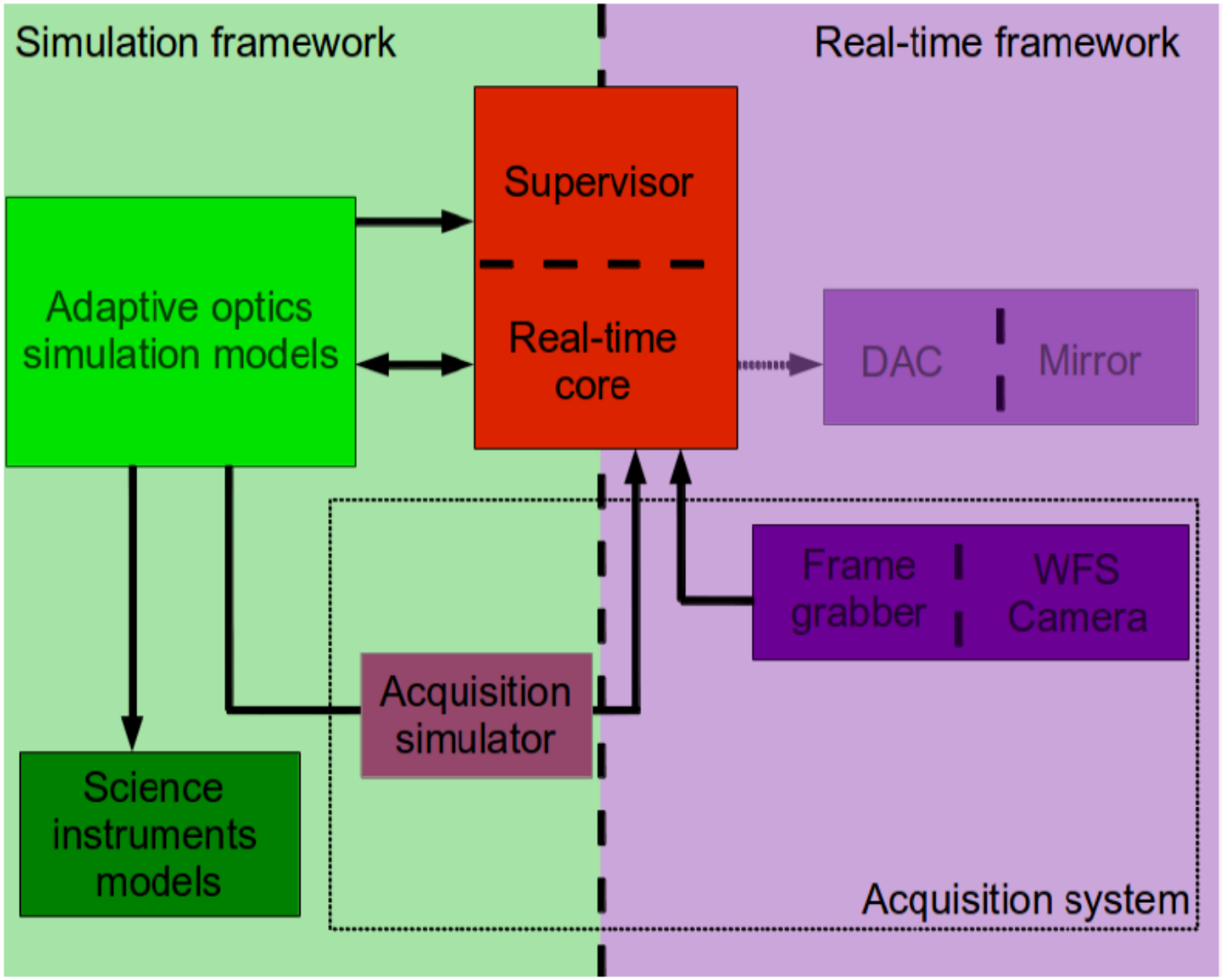


Illustration 1: COMPASS framework

Projet COMPASS

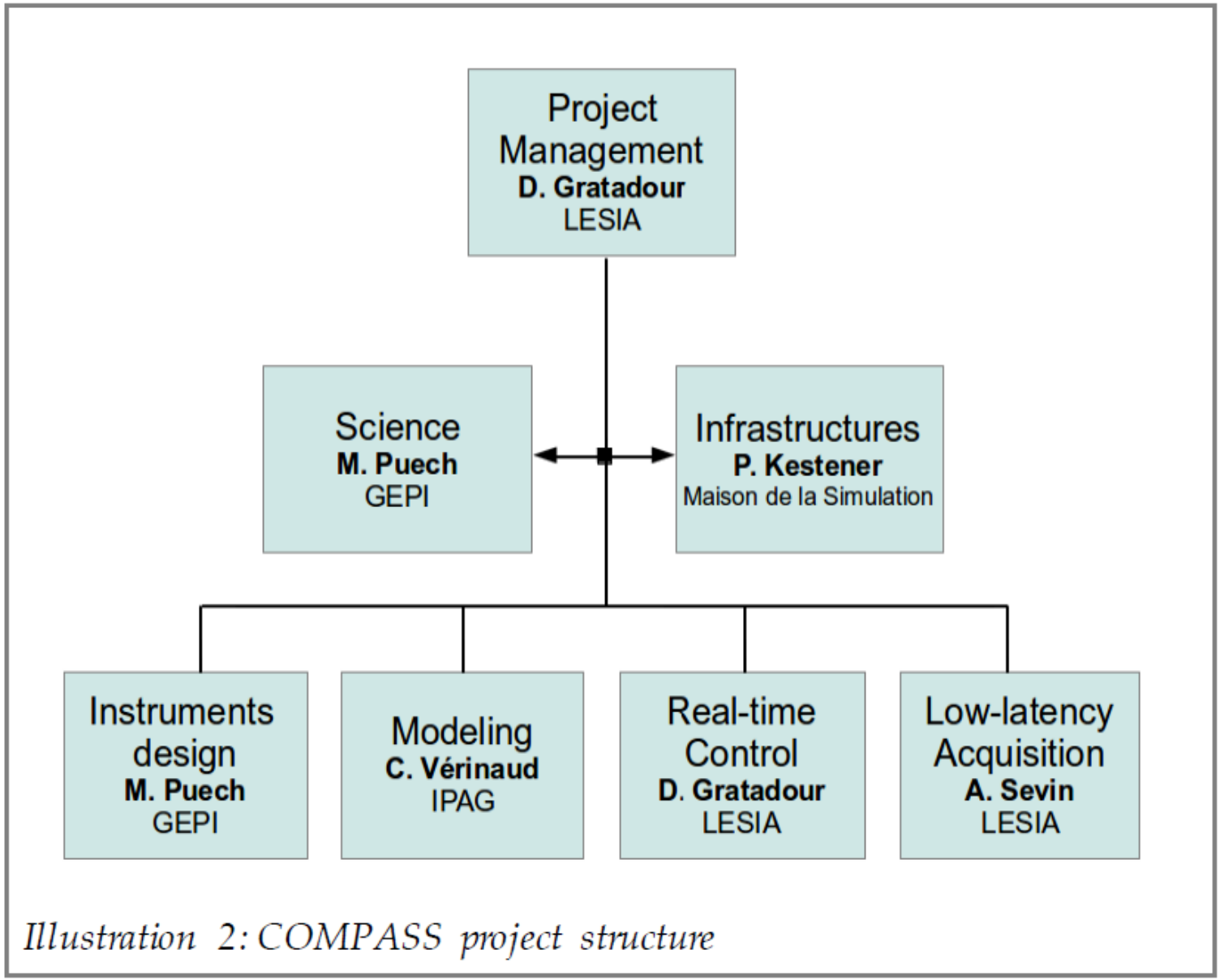


Illustration 2: COMPASS project structure

WBS	Name	Work
1	**Project Management**	**2080d**
1.1	General management	520d
1.2	Dissemination strategies	520d
1.3	Science working group	520d
1.4	Computing working group	520d
2	**Science**	**135d**
2.1	Interfaces to instruments models	45d
2.2	Preparatory studies	90d
3	**Infrastructures**	**1239d**
3.1	Software framework management	520d
3.2	Maintenance plan & procedures	45d
3.3	Platform procurement & deployment	154d
3.4	MAOS compatibility	520d
4	**Modeling**	**390d**
4.1	Deformable mirror model	195d
4.2	Pyramid model	195d
5	**Real-time control**	**1820d**
5.1	Least Square & Minimum variance	45d
5.2	Kalman filter	260d
5.3	Tomography : Learn & Apply	220d
5.4	Iterative methods (FRiM)	255d
5.5	Supervisor	520d
5.6	Advanced methods (Fourier), new devel.	520d
6	**Low-latency acquisition**	**440d**
6.1	Acq. simulator	45d
6.2	Frame grabber impl. & acq. prototype	395d
7	**Instrument design**	**515d**
7.1	ELT-CAM	125d
7.2	ELT-IFU / ELT-MOS	260d
7.3	ELT-PF	130d
8	**Milestones**	
8.1	Kick-off	N/A
8.2	Development phase kick-off	N/A
8.3	Mid-term review	N/A
8.4	Final review	N/A

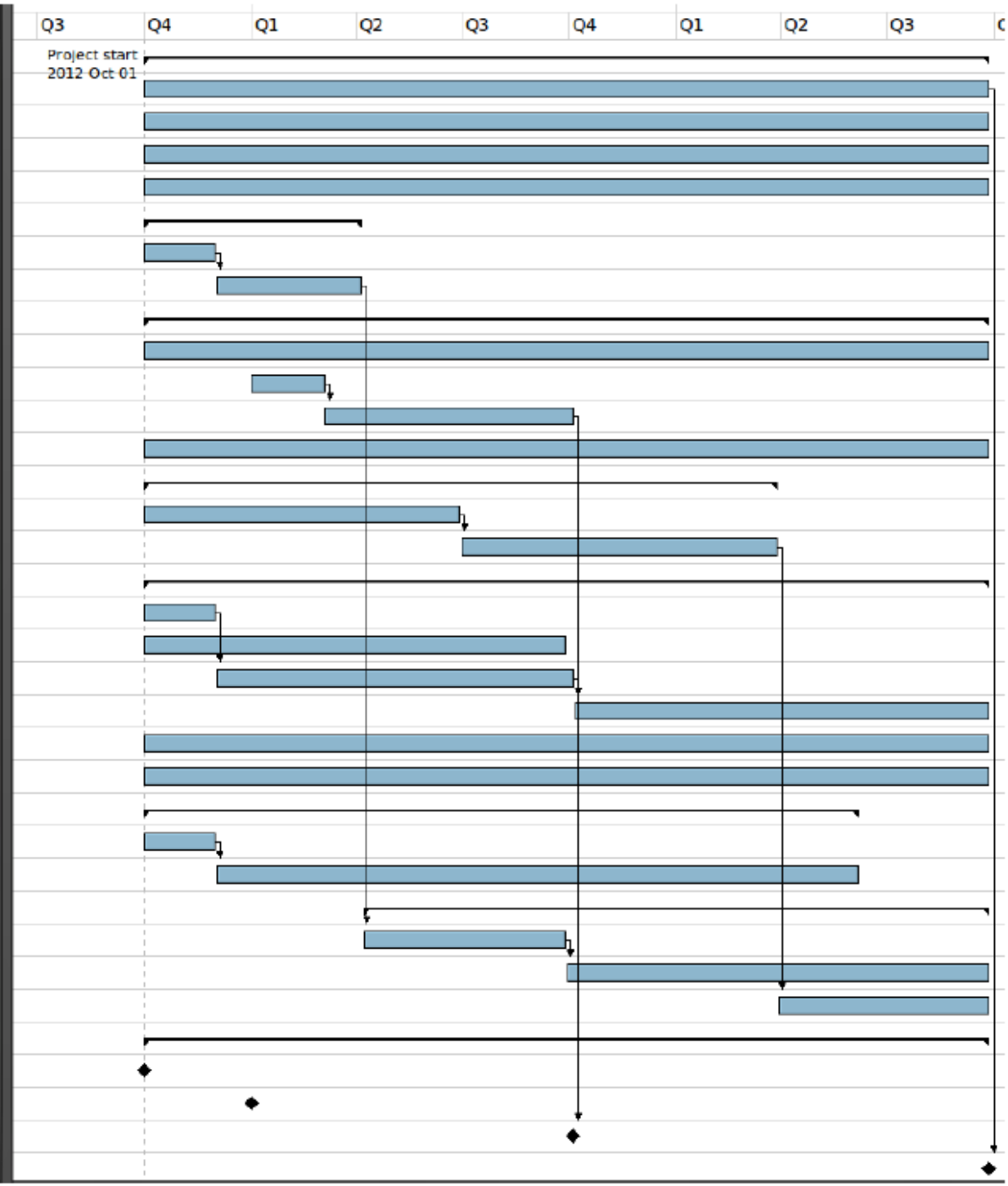


Illustration 2: Gantt diagram for the COMPASS project

Projet COMPASS

ID	Title & description	Date
M1	**COMPASS kick-off**	T0
M2	**COMPASS development phase kick-off** At this point, hardware & software platform are defined and maintenance plan is in place as well as the collaborative development framework and wiki	T0+3m
M3	**Mid-term review: COMPASS framework validation** At this point SWG and CWG will organize a mid-term reviews of both science (task 4 & 7) and computing (task 5 & 6) related activities and the overall platform should be deployed at each partner institute (*D3.3: COMPASS readiness review*)	T0+12m
M4	**COMPASS final review**	T0+24m

Projet COMPASS

- **Ressources**
  - Effort total : 260 personnes.mois sur 24 mois
  - 120 personnes.mois demandées à l'ANR
  - Coût total : ~2.5M€
  - Dont, demandés à l'ANR : ~800k€
  - Le LESIA est le contributeur principal avec un effort de 60 personnes.mois sur 24 mois