

WGNE39 + WGNE/WGSIP Joint Meeting | Toulouse, 4 Nov 2024



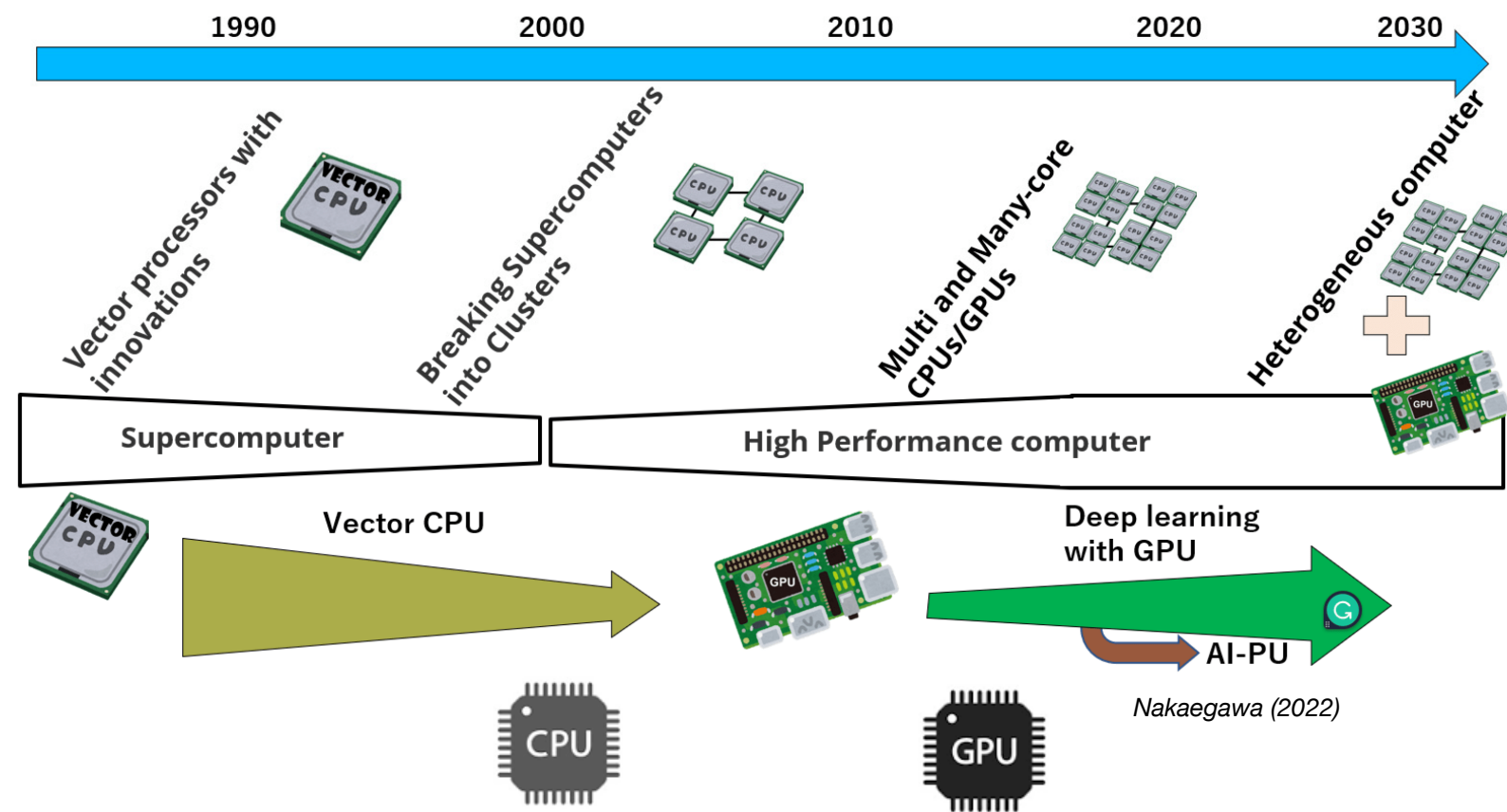
COMputing PARadigms towards efficient, modular and trainable CLimate Models

Julien Le Sommer

with : S. Valcke, Y. Meurdesoif, T. Dubos, P. Rampal

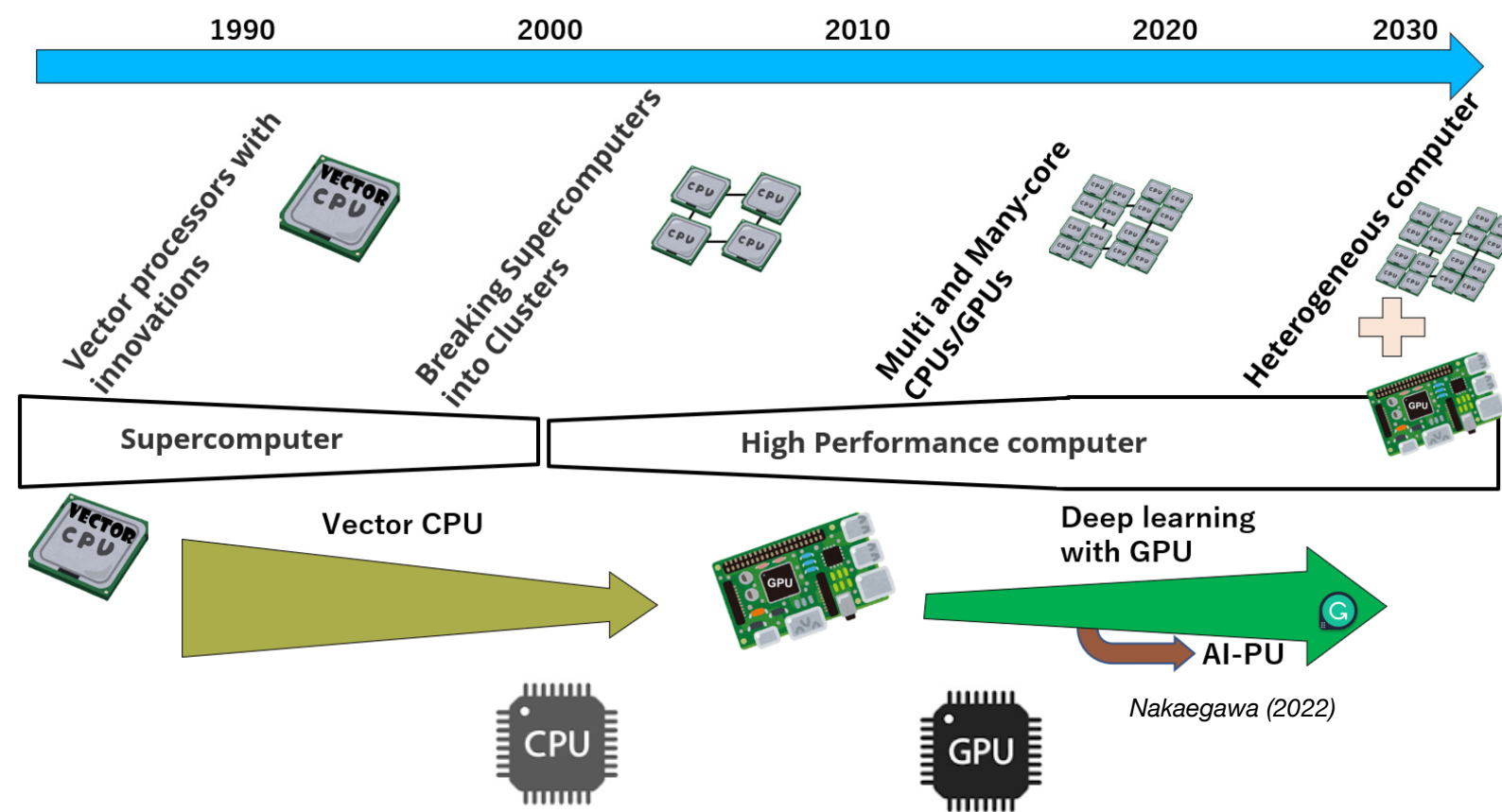
Specific drivers for COMPACT project

Specific drivers for COMPACT project

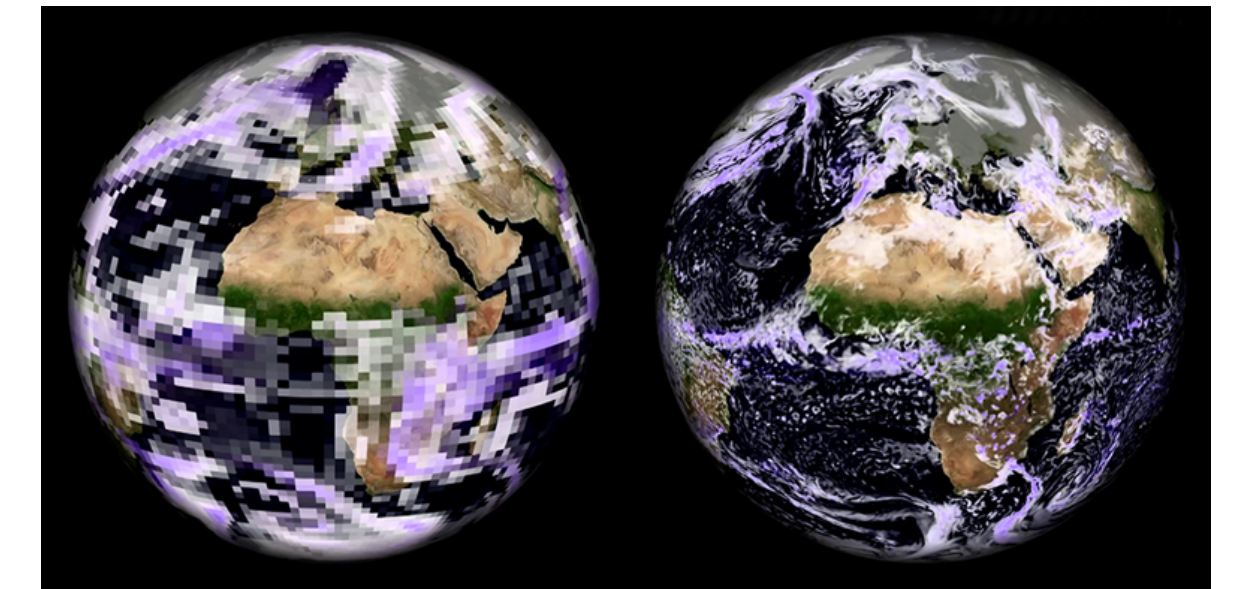


Heterogeneous
computing architectures

Specific drivers for COMPACT project

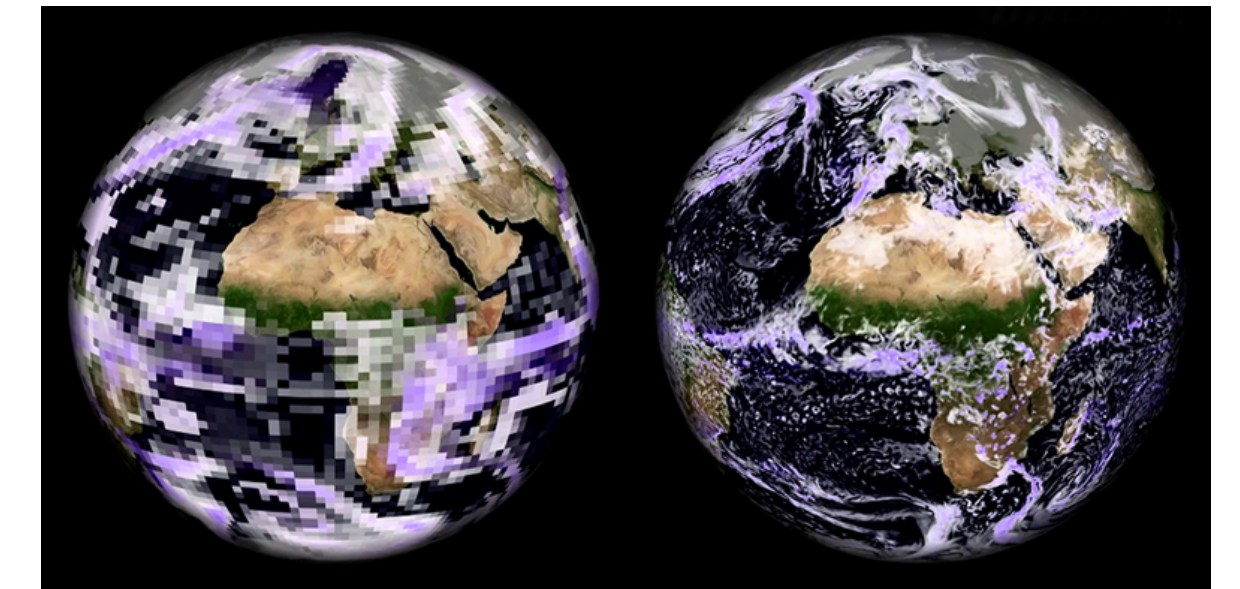
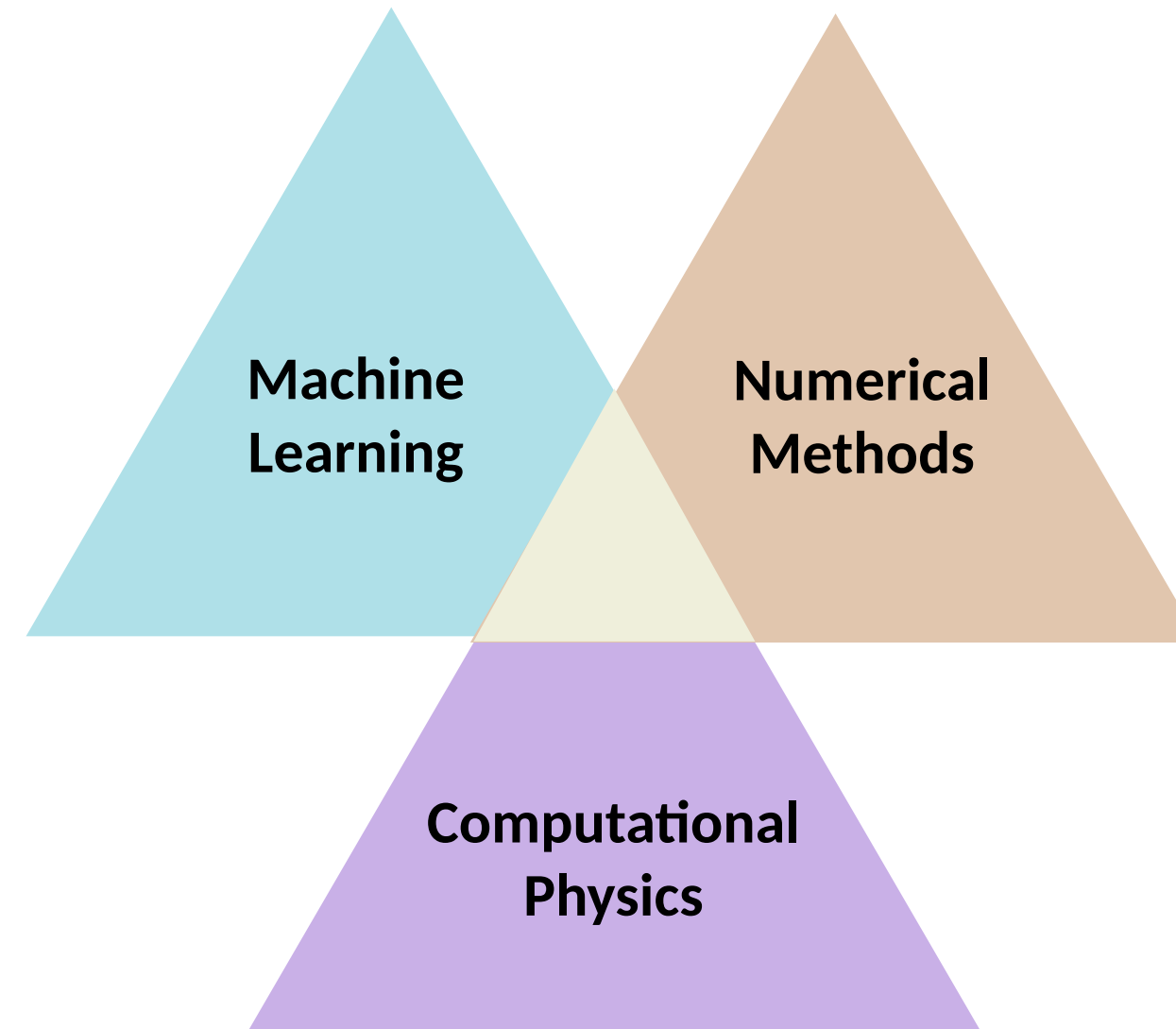
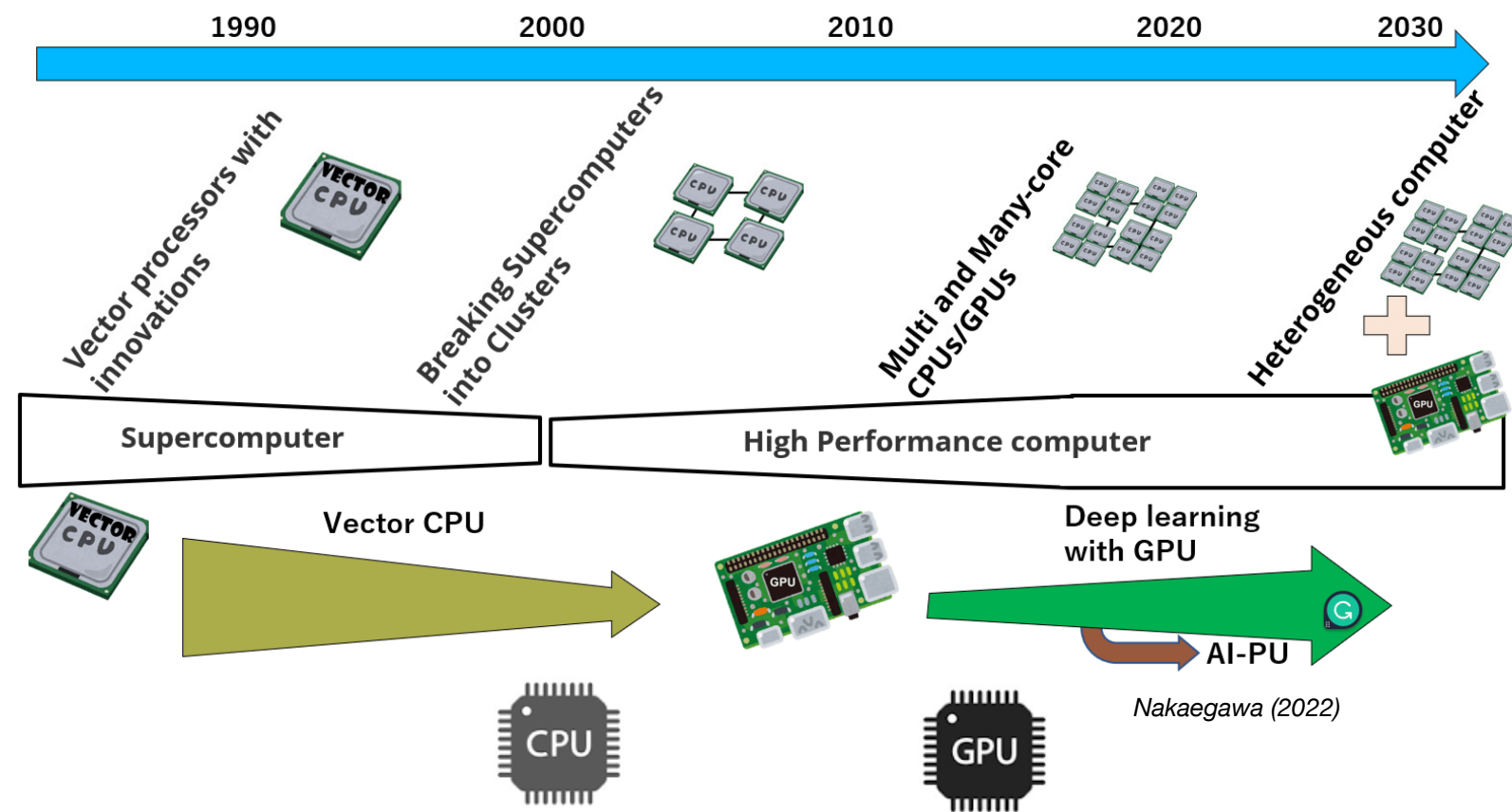


Heterogeneous computing architectures



New usages of models with climate services

Specific drivers for COMPACT project

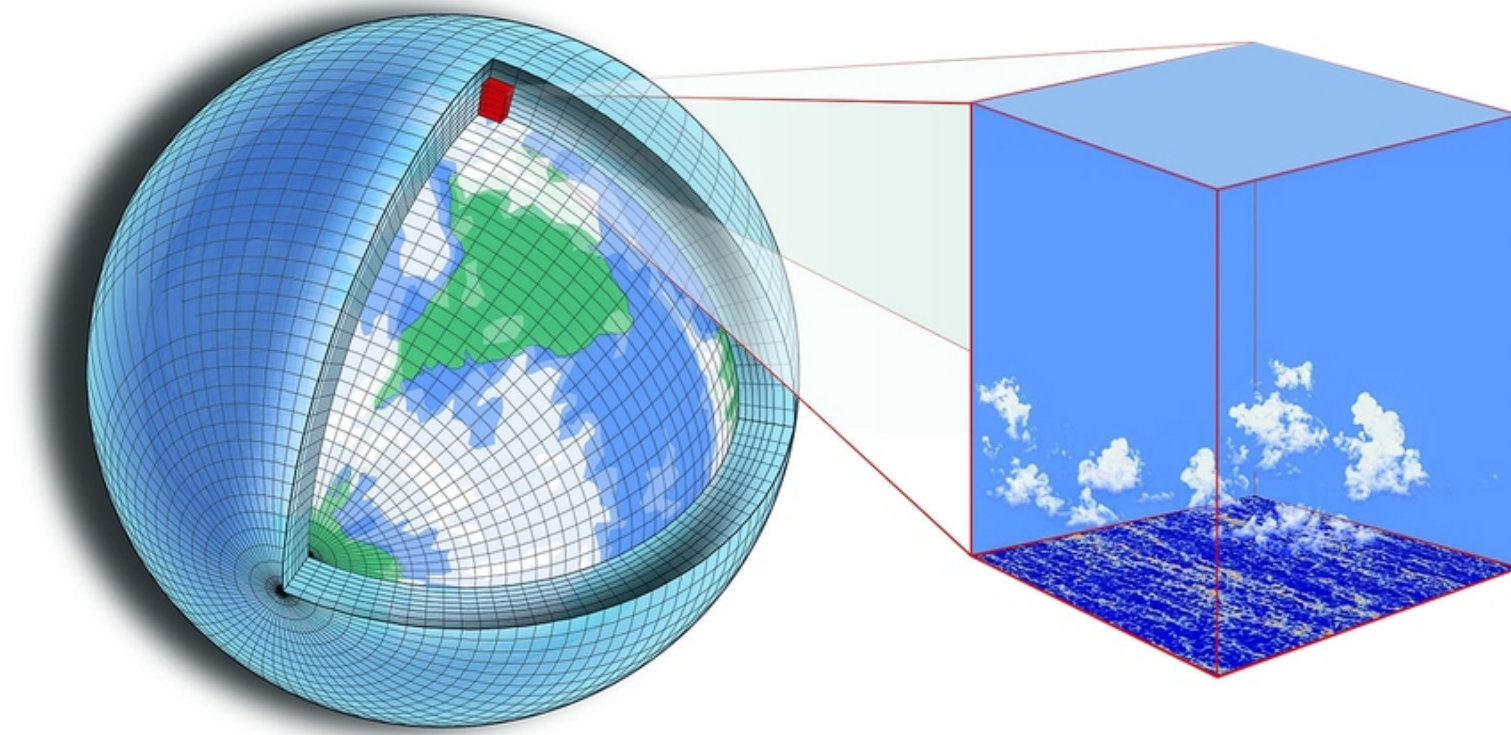


Heterogeneous computing architectures

Emerging paradigms in scientific computing

New usages of models with climate services

Ambition and positioning of COMPACT



ESMs as software systems

Ambition and positioning of COMPACT

```
def __init__(self):  
    self.file = None  
    self.fingerprints = set()  
    self.logdupes = True  
    self.debug = debug  
    self.logger = logging.getLogger(__name__)  
    if path:  
        self.file = open(os.path.join(path, "requests.log"),  
                        "a")  
        self.file.seek(0)  
        self.fingerprints.update(set(requests))  
@classmethod  
def from_settings(cls, settings):  
    debug = settings.getbool("REQUESTS_DEBUG")  
    return cls(job_dir(settings), debug)  
def request_seen(self, request):  
    fp = self.request_fingerprint(request)  
    if fp in self.fingerprints:  
        return True  
    self.fingerprints.add(fp)  
    if self.file:  
        self.file.write(fp + os.linesep)  
def request_fingerprint(self, request):  
    return request_fingerprint(request)
```



ESMs as software systems

Ambition and positioning of COMPACT

```
def __init__(self):
    self.file = None
    self.fingerprints = set()
    self.logdupes = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(path, 'requests.log'),
                        'a')
        self.file.seek(0)
        self.fingerprints.update(self.request_fingerprints())

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('REQUEST_FINGERPRINT')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

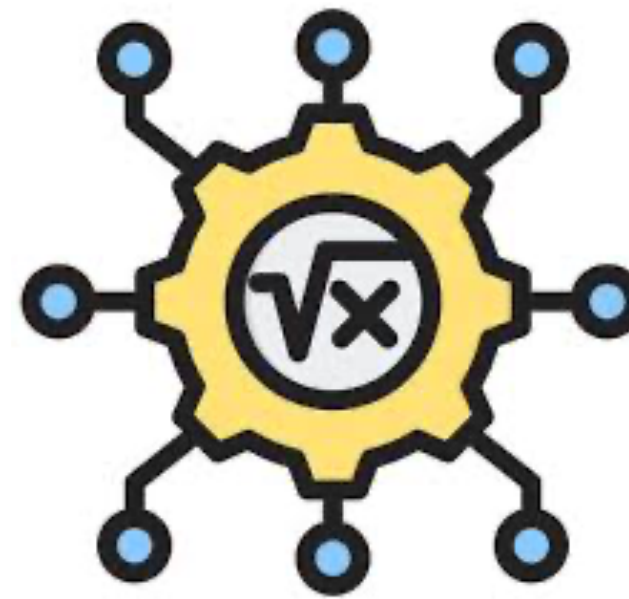


ESMs as software systems

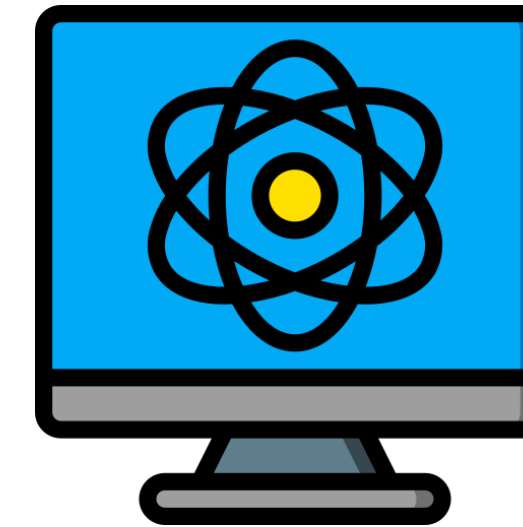
- ▶ gradual evolution of systems
long term engineer positions
- ▶ higher risk / reward activities
through PhDs / research projects

Ambition and positioning of COMPACT

```
def __init__(self):  
    self.file = None  
    self.fingerprints = set()  
    self.logdupes = True  
    self.debug = debug  
    self.logger = logging.getLogger(__name__)  
    if path:  
        self.file = open(os.path.join(path, 'requests.log'),  
                        'a')  
        self.file.seek(0)  
        self.fingerprints.update(requests) for r in self.fingerprints:  
            if r in requests:  
                continue  
            self.fingerprints.add(r)  
            self.file.write(r + os.linesep)  
@classmethod  
def from_settings(cls, settings):  
    debug = settings.getbool('debug', False)  
    return cls(job_dir(settings), debug)  
def request_seen(self, request):  
    fp = self.request_fingerprint(request)  
    if fp in self.fingerprints:  
        return True  
    self.fingerprints.add(fp)  
    if self.file:  
        self.file.write(fp + os.linesep)  
def request_fingerprint(self, request):  
    return request_fingerprint(request)
```



Applied maths



Computer science

evidence / science
based decisions

ESMs as software systems

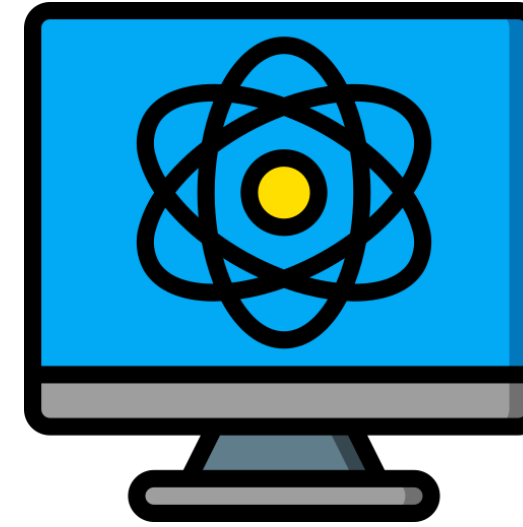
- ▶ gradual evolution of systems
long term engineer positions
- ▶ higher risk / reward activities
through PhDs / research projects

Ambition and positioning of COMPACT

```
def __init__(self):  
    self.file = None  
    self.fingerprints = set()  
    self.logdupes = True  
    self.debug = debug  
    self.logger = logging.getLogger(__name__)  
    if path:  
        self.file = open(os.path.join(path, 'requests.log'),  
                        'a')  
        self.file.seek(0)  
        self.fingerprints.update(requests) for r in self.requests:  
    @classmethod  
    def from_settings(cls, settings):  
        debug = settings.getbool('debug', True)  
        return cls(job_dir(settings), debug)  
    def request_seen(self, request):  
        fp = self.request_fingerprint(request)  
        if fp in self.fingerprints:  
            return True  
        self.fingerprints.add(fp)  
        if self.file:  
            self.file.write(fp + os.linesep)  
    def request_fingerprint(self, request):  
        return request_fingerprint(request)
```



Applied maths



Computer science

evidence / science based decisions

community / open science driven

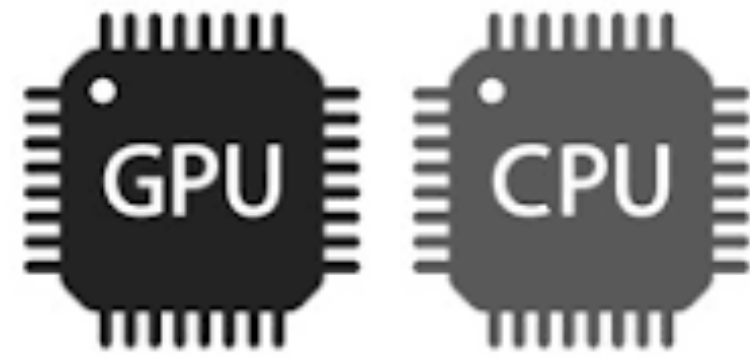
ESMs as software systems

- ▶ gradual evolution of systems
long term engineer positions
- ▶ higher risk / reward activities
through PhDs / research projects



A 8-years roadmap

A 8-years roadmap



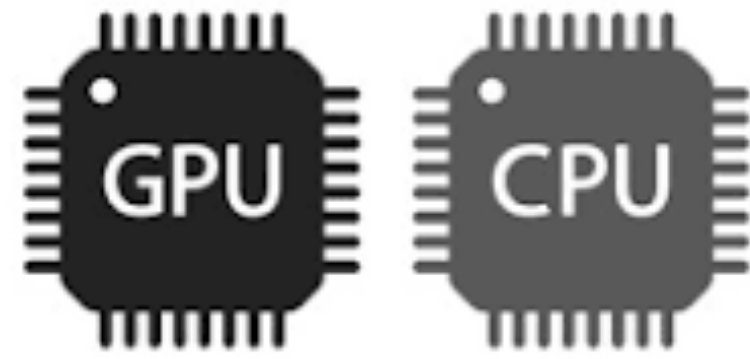
porting codes to GPUs



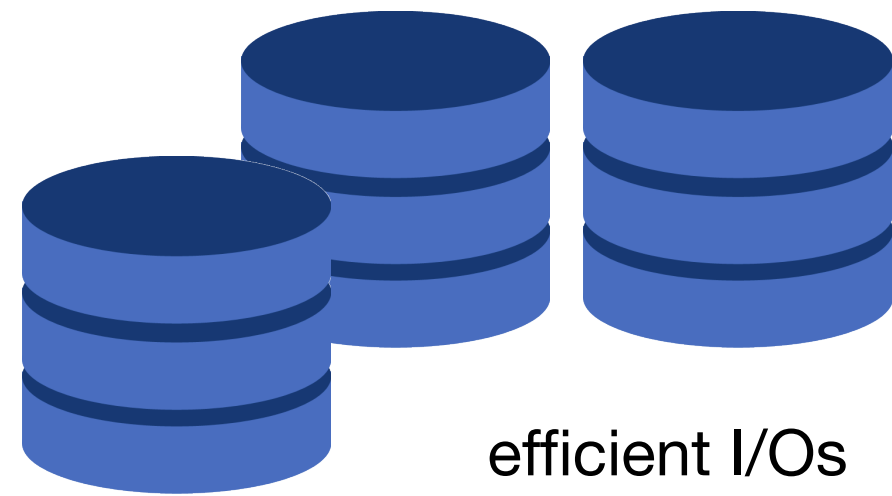
efficient I/Os

Computational **efficiency**
and portability

A 8-years roadmap

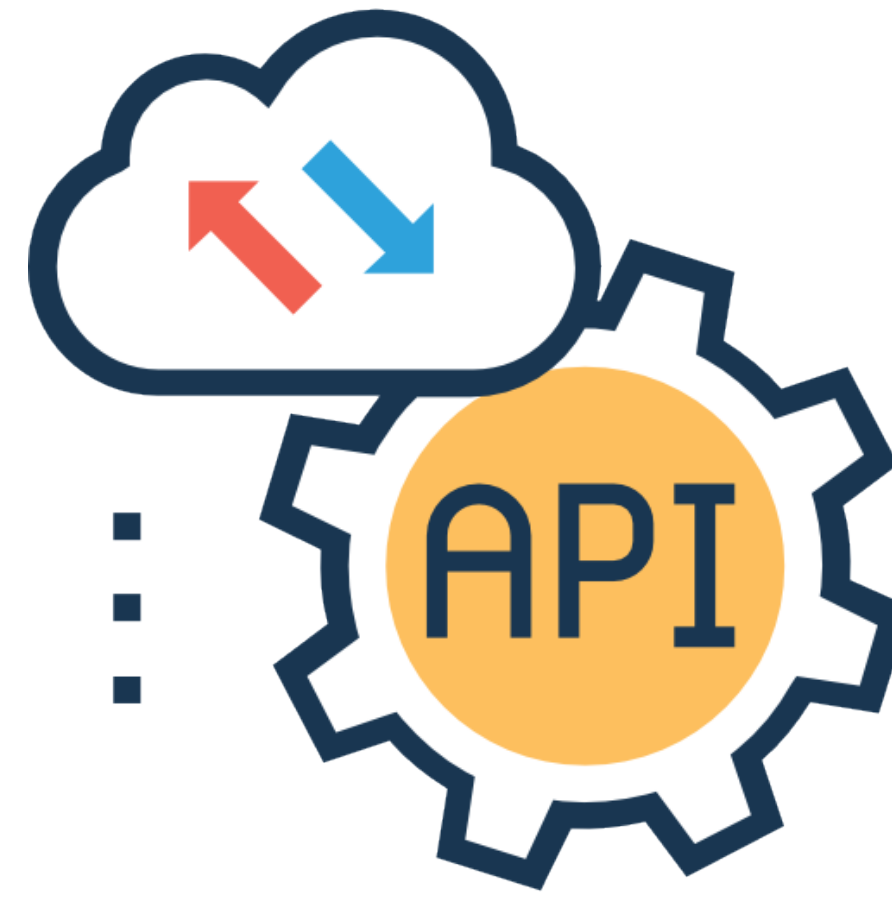


porting codes to GPUs



efficient I/Os

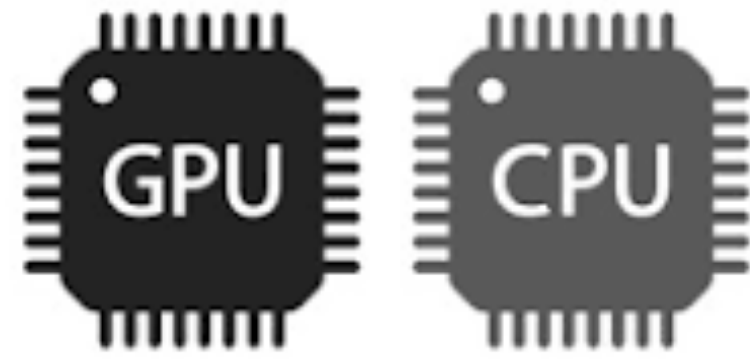
Computational **efficiency**
and portability



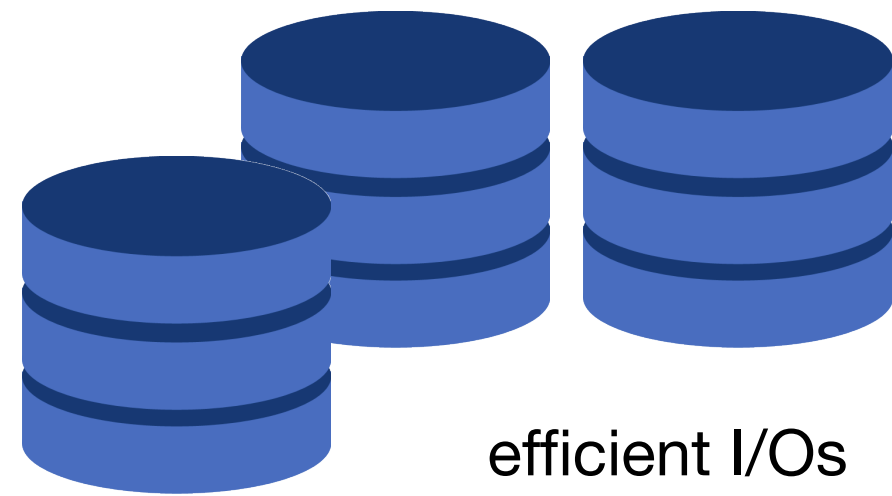
internal interfaces
across subcomponents

Modularity, APIs and
system-wide design

A 8-years roadmap

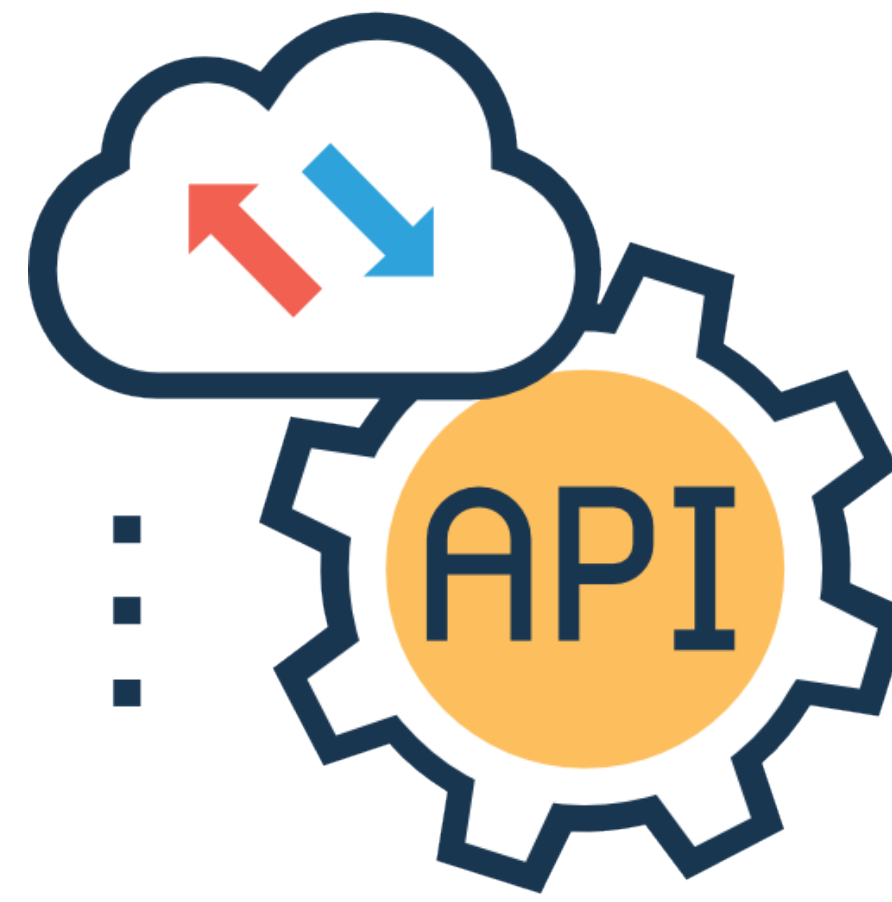


porting codes to GPUs



efficient I/Os

Computational **efficiency**
and portability

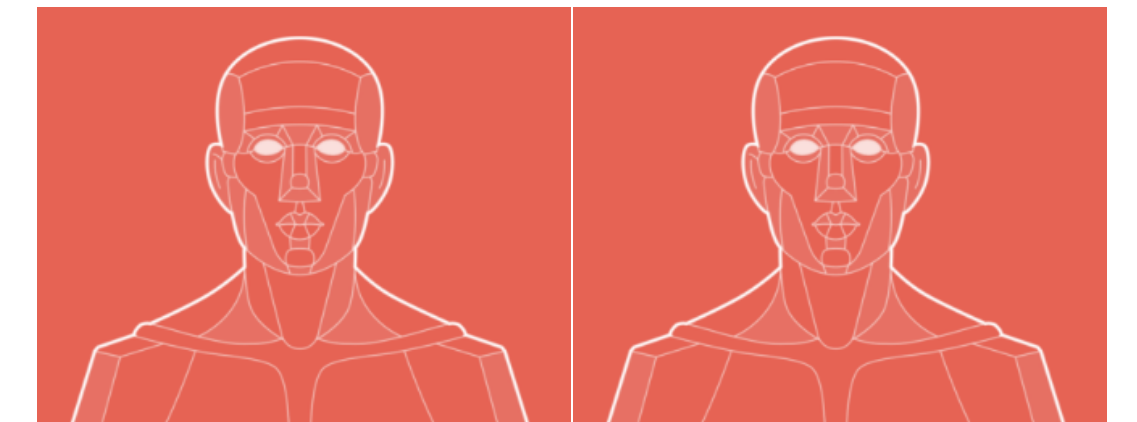


internal interfaces
across subcomponents

Modularity, APIs and
system-wide design



protocols for online inference



neural emulation of components

AI-readiness, emulation
and differentiability