International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 1s, 2025
https://theaspd.com/index.php

Sustainable Computing Infrastructure: Designing Energy Efficient Data Centers and Computing Infrastructure

Dr. Sweety Gokul Mahajan[1], Dr. Manoj Damodar Salunke[2], Prof. Firoza Mirza[3], Mr Gokul Vasant Mahajan[4], Nilam M Deshmukh[5]
[1]Assistant Professor, Department of Computer Engineering, Guru Gobind Singh College of Engineering and Research Centre Nashik, sweety.3288@gmail.com
[2]Assistant Professor, Automation and Robotics Department, Guru Gobind Singh College of Engineering and Research Centre Nashik, manojsalunke2012@gmail.com
[3]Head, Training & Placement Department/ Assistant Professor – MCA, St.Wilfreds College of Computer Sciences, firozamirza.stwc@gmail.com
[4]Assistant Professor & TPO, Shatabdi Institute of Engineering and Research, Nashik
gokul.sier@gmail.com
[5]Assistant Professor, Department of Computer Engineering, Guru Gobind Singh College of Engineering and Research Centre Nashik, nilam.deshmukh@ggsf.edu.in

Abstract
Data centers and computing infrastructure underpin the digital economy but consume rapidly growing amounts of electricity, water, and materials. This paper synthesizes state-of-the-art design strategies and operational practices to achieve energy and resource efficient, low carbon computing at hyper scale and edge. We unify facility engineering, IT architecture, and workload orchestration into a holistic framework that targets power usage effectiveness (PUE) ≤ 1.20, water usage effectiveness (WUE) < 0.2 L/kWh, and near real time carbon optimization (24/7 CFE matching). We review thermal envelopes and air/liquid cooling guidance, power chain efficiency, renewable and storage integration, and carbon aware scheduling. We propose a practical design reference architecture and a multi objective optimization methodology that balances efficiency, resilience, latency, and cost. Case evidence and modeling show that combining high return measures–rightsizing, advanced airflow management, heat reuse, free cooling, direct to chip liquid cooling, and carbon aware scheduling–can reduce facility energy overhead by 30–60%, IT energy by 10–25%, and Scope 2 emissions by >70% in grids with high variable renewables. We conclude with a roadmap and research agenda for AIera workloads.
Keywords: *data center efficiency, PUE, WUE, CUE, liquid cooling, carbon aware scheduling, grid interactive efficient buildings (GEB), renewable energy, edge computing, circular economy*

1. INTRODUCTION

Demand for compute is expanding due to cloud adoption, AI/ML training and inference, streaming, and IoT. Projections indicate data center electricity use could more than double by 2030, intensifying policy and grid pressures. Designing sustainable computing infrastructure is therefore a technical, economic, and environmental imperative. While industry standard metrics such as PUE have driven progress, holistic optimization must encompass IT efficiency, power and cooling, water stewardship, carbon intensity, and embodied impacts across the full lifecycle.[1]

1.1 Contributions

This paper:
1) Consolidates best practices across facility and IT layers into a design reference architecture for sustainable computing infrastructure.
2) Provides a methodological toolkit for multi objective planning (energy, water, carbon, reliability, latency, cost) with actionable targets and KPIs.
3) Details control strategies for carbon aware and gridinteractive operations compatible with AI/ML workloads.
4) Identifies research gaps in thermal management for high density racks, AI scheduling, and circularity.[3]

2. Background and Metrics

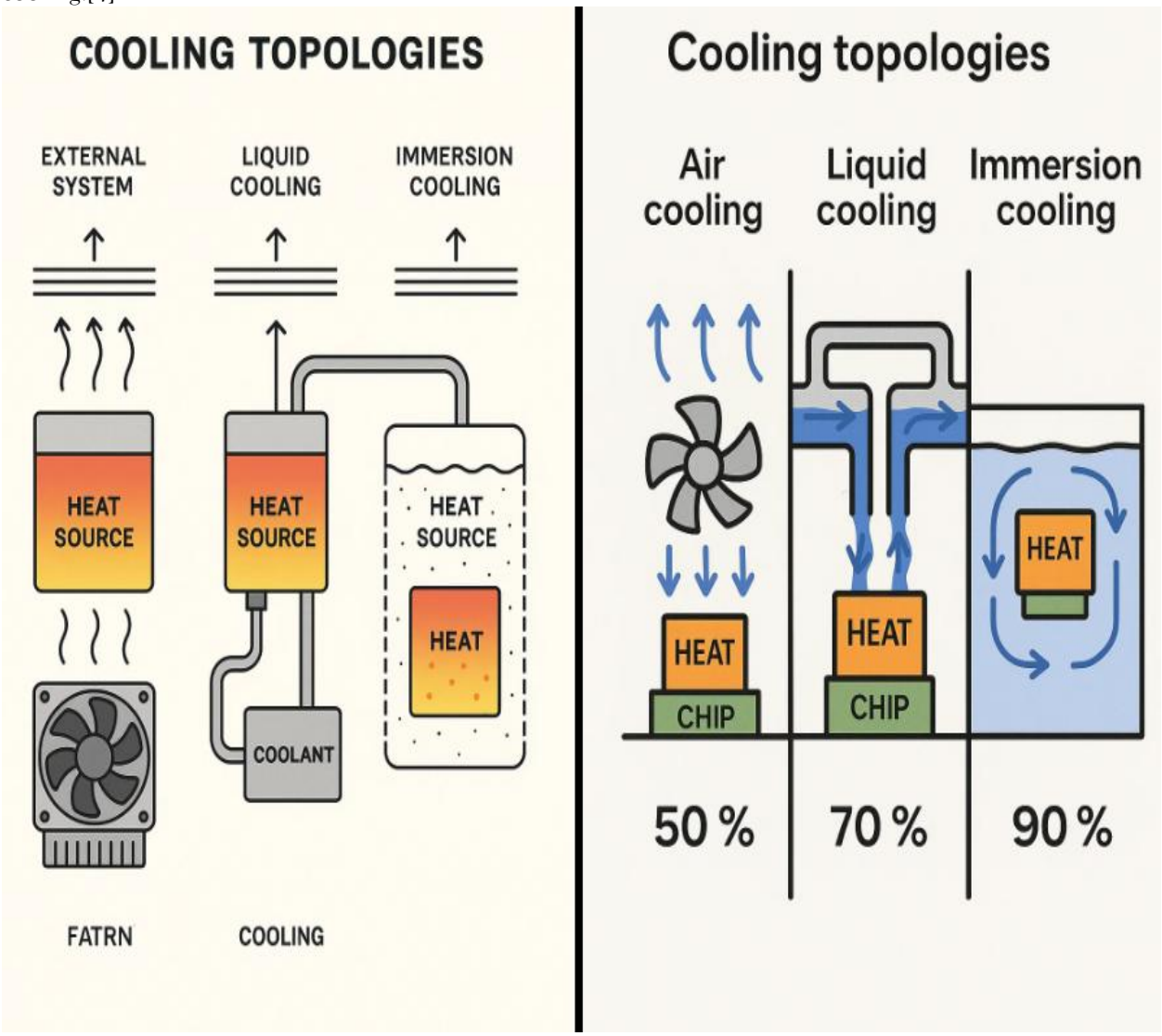
2.1 Energy and resource metrics

- **Power Usage Effectiveness (PUE):** ratio of total facility power to IT equipment power; lower is better.
- **Water Usage Effectiveness (WUE):** liters of water per kWh of IT energy.

- **Carbon Usage Effectiveness (CUE):** kg CO_2e per kWh of IT energy (or per kWh delivered at rack), tied to grid emissions and procurement.
- **Energy Reuse Effectiveness (ERE):** credits useful heat recovery.
- **Server Utilization / IT Efficiency:** perf/W (e.g., SPECpower), consolidation rate, virtualization, accelerator efficiency.

2.2 Thermal envelopes and classes

Adherence to ASHRAE TC 9.9 guidelines enables widening temperature and humidity ranges to unlock free cooling and liquid strategies while protecting reliability. The A1–A4 classes define air cooled allowable and recommended envelopes; classes H1+ address high density equipment and liquid cooling.[4]



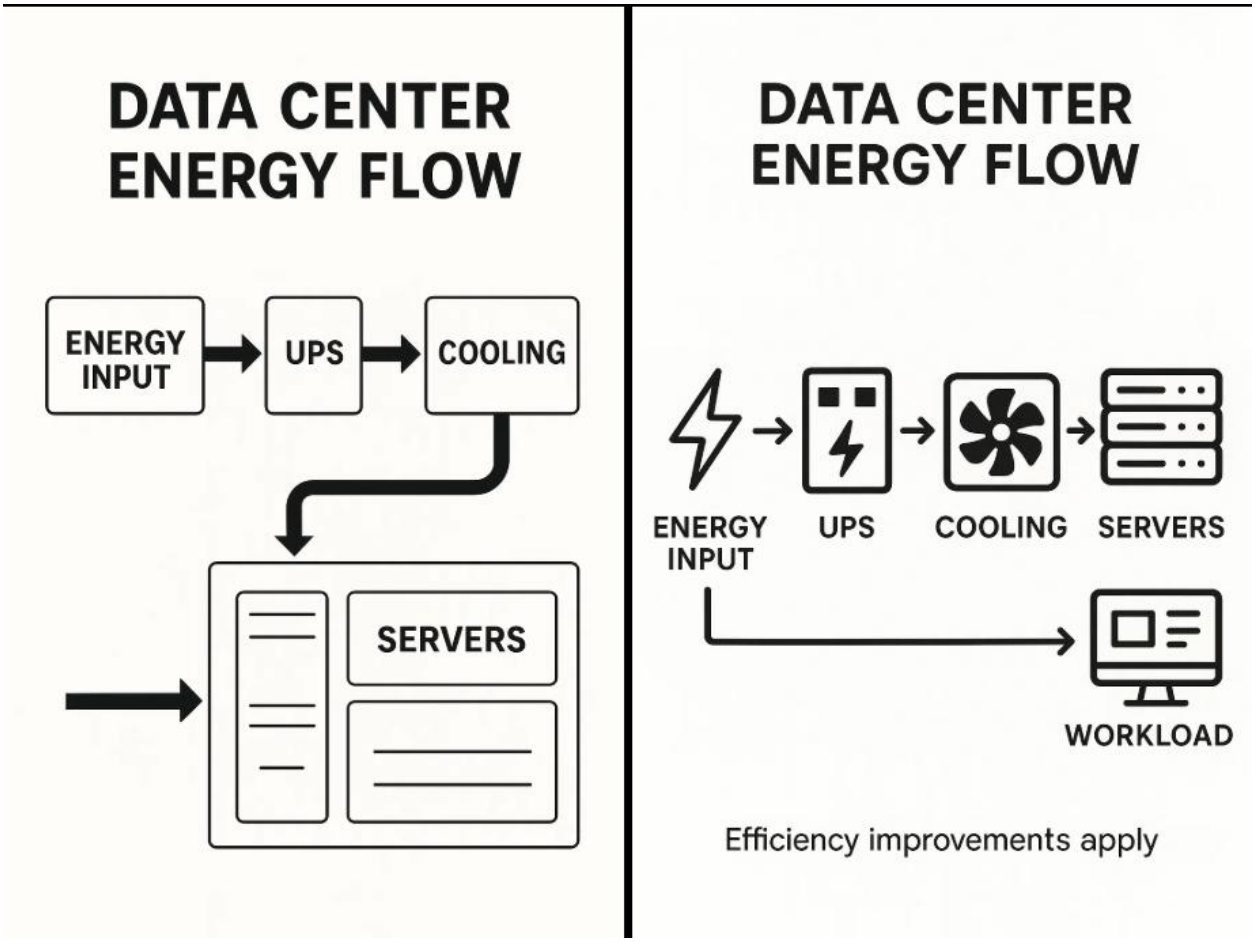
2.3 Workload classes

- **Latency sensitive:** real-time analytics, trading, gaming, conversational inference—favor edge or low latency zones.
- **Throughput or batch:** training, rendering, ETL—flexible in time and location; prime candidates for carbon aware shifting.[2]
- **Storage centric:** object/file/block—optimize for media power and data placement.

3. Design Principles and Reference Architecture

We propose a layered architecture integrating site selection, building/facility engineering, power chain, thermal systems, whitespace layout, IT hardware, and orchestration.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 1s, 2025
https://theaspd.com/index.php



4. Methods: Multi-Objective Planning and Control

4.2 Optimization formulation

Below is an example Python snippet to model a simplified multi objective optimization problem for data center operations:[5]

```
import pulp

# Decision variables: supply temp, server utilization, renewable share
T_supply = pulp.LpVariable('T_supply', lowBound=18, upBound=30)
utilization = pulp.LpVariable('utilization', lowBound=0.2, upBound=0.9)
renew_share = pulp.LpVariable('renew_share', lowBound=0, upBound=1)

# Objective weights
alpha, beta, gamma = 0.5, 0.3, 0.2

# Objective: minimize energy + water + carbon proxy
model = pulp.LpProblem('DataCenterOpt', pulp.LpMinimize)
model += alpha * (1000/utilization) + beta * (T_supply/25) + gamma * (1-renew_share)

# Constraints (example)
model += utilization >= 0.5
model += T_supply <= 27
model.solve()
print("Optimal Supply Temp:", T_supply.varValue)
print("Optimal Utilization:", utilization.varValue)
print("Renewable Share:", renew_share.varValue)
```

This illustrative code shows how optimization variables (temperature setpoint, server utilization, renewable share) can be tuned to balance energy, water, and carbon objectives.

6. Carbon Aware and Grid Interactive Operations
6.1 Example scheduling simulation
A simplified pseudo program for carbon aware scheduling:

```
import random

jobs = [f"job_{i}" for i in range(1, 11)]
carbon_intensity = {h: random.randint(200, 600) for h in range(24)}  # gCO2/kWh

schedule = {}
for job in jobs:
    best_hour = min(carbon_intensity, key=carbon_intensity.get)
    schedule[job] = best_hour
    carbon_intensity[best_hour] += 50  # simulate load effect

print("Carbon-aware schedule:")
for job, hour in schedule.items():
    print(job, "->", hour, "h")
```

This demonstrates shifting batch workloads into hours with the lowest carbon intensity.

Appendix A: Example Control Policies (Pseudo Code)
A1. Carbon aware scheduling weight update

```
Every hour h:
  ci[h]    = grid_marginal_carbon_intensity(region, h)
  price[h] = energy_price(region, h)
  temp[h]  = ambient_forecast(h)
  w[h]     = α*(ci[h]/max(ci)) + β*(price[h]/max(price)) + γ*(temp[h]/max(temp))
  update_queue_priorities(batch_jobs, weight=w[h])
```

A2. MPC setpoint optimization

```
At 5min intervals:
  state = (rack_loads, ΔT, valve_positions, SoC_BESS)
  predict PUE, WUE, CUE over horizon H using digital twin
  solve min α·Energy + β·Water + γ·Carbon + penalties for SLA/risk
  apply optimal (T_supply_air, T_supply_water, fan_speed, pump_speed)
```

Appendix B: Template Bill of Materials (BOM)

- High efficiency UPS (≥97–99%); MV switchgear; busway
- CRAH/CRAC with EC fans; dry coolers with adiabatic assist; CDUs
- Direct to chip cold plates; reardoor HEX; immersion tanks (as needed)
- Rack PDUs with peroutlet metering; leak detection; sensors
- BESS; PV inverters; microgrid controller
- Telemetry stack (power, thermal, IT); data historian; analytics

Reference Implementations (Code)
C1. Python: PUE/WUE/CUE Calculator & Reporter (CLI)

```
#!/usr/bin/env python3
"""Compute PUE, WUE, CUE from metered data and emit a daily report."""
import argparse, csv, json, statistics, sys
from datetime import datetime

def read_rows(path):
    with open(path, newline='') as f:
        for r in csv.DictReader(f):
            yield {k: (float(v) if v not in (None, '', 'NaN') else None) for k,v in r.items()}

def compute_kpis(rows):
```

```
    pue = []
    wue = []
    cue = []
    for r in rows:
        it_kwh   = r.get('it_kwh')
        total_kwh= r.get('total_kwh')
        water_l  = r.get('water_l')
        kg_co2   = r.get('kg_co2')
        if it_kwh and total_kwh and it_kwh>0:
            pue.append(total_kwh/it_kwh)
        if it_kwh and water_l is not None and it_kwh>0:
            wue.append(water_l/it_kwh)
        if it_kwh and kg_co2 is not None and it_kwh>0:
            cue.append(kg_co2/it_kwh)
    def stats(x):
        if not x: return None
        return {
            'avg': sum(x)/len(x),
            'p50': statistics.median(x),
            'p95': statistics.quantiles(x, n=20)[18],
            'min': min(x),
            'max': max(x)
        }
    return {'PUE': stats(pue), 'WUE': stats(wue), 'CUE': stats(cue)}

if __name__ == '__main__':
    ap = argparse.ArgumentParser()
    ap.add_argument('--input', required=True, help='CSV with columns: timestamp,it_kwh,total_kwh,wa
ter_l,kg_co2')
    ap.add_argument('--out', default='kpis.json')
    args = ap.parse_args()
    kpis = compute_kpis(list(read_rows(args.input)))
    kpis['generated_at'] = datetime.utcnow().isoformat()
    with open(args.out, 'w') as f: json.dump(kpis, f, indent=2)
    print(json.dumps(kpis, indent=2))
```

Sample CSV header

```
timestamp,it_kwh,total_kwh,water_l,kg_co2
```

C2. Python: Carbon Intensity Aware Batch Scheduler (Simulated)

```
#!/usr/bin/env python3
"""Toy scheduler that shifts flexible jobs to lowcarbon hours/regions."""
from dataclasses import dataclass
from typing import List, Dict

@dataclass
class Region:
    name: str
    hourly_ci: List[float]  # gCO2/kWh, length 24
    capacity: List[int]     # slots available each hour

@dataclass
class Job:
    id: str
    hours: int
    latest_finish: int  # hour index 0..47
```

```
regions = [
    Region('North', [600,550,500,480,460,420,380,350,340,330,320,310,
                     300,310,320,350,400,450,500,520,540,560,580,610], [10]*24),
    Region('South', [500,480,450,430,400,360,320,300,280,260,250,240,
                     230,240,250,270,320,380,420,450,470,490,510,530], [8]*24)
]

jobs = [Job(f'job{i}', hours=4, latest_finish=24+i) for i in range(8)]

schedule: Dict[int, Dict[str, List[str]]] = {h:{r.name:[] for r in regions} for h in range(48)}

for j in jobs:
    # greedy: pick consecutive hours with lowest CI across regions before deadline
    choices = []
    for start in range(0, j.latest_finish - j.hours + 1):
        cost=0; feasible=True; picks=[]
        for k in range(j.hours):
            h = start+k
            # choose best region this hour with capacity
            best = None
            best_ci = float('inf')
            for r in regions:
                if h%24 < len(r.capacity) and r.capacity[h%24] > 0:
                    ci = r.hourly_ci[h%24]
                    if ci < best_ci:
                        best_ci = ci; best = r
            if best is None:
                feasible=False; break
            cost += best.hourly_ci[h%24]
            picks.append((h,best))
        if feasible:
            choices.append((cost, picks))
    if not choices:
        raise RuntimeError(f'No window for {j.id}')
    _, picks = min(choices, key=lambda x: x[0])
    for h,r in picks:
        schedule[h][r.name].append(j.id)
        r.capacity[h%24] -= 1

# print compact schedule
for h in range(0, 24):
    row = {r.name: len(schedule[h][r.name]) for r in regions}
    print(f'h{h:02d}: {row}')
```

C3. Python: Simple Thermal Controller (PI + feedforward)

```
class ThermalLoop:
    def __init__(self, k_p=0.6, k_i=0.2, ff=0.1, t_supply_min=18.0, t_supply_max=27.0):
        self.k_p=k_p; self.k_i=k_i; self.ff=ff
        self.integral=0.0
        self.tmin=t_supply_min; self.tmax=t_supply_max
    def step(self, t_return, rack_kw, t_target=24.0, dt=60.0):
        error = t_target - t_return
        self.integral += error*dt
        # feedforward lift based on load
        lift = self.ff * rack_kw
```

```
        t_supply = t_return - (self.k_p*error + self.k_i*self.integral) - lift
        return max(self.tmin, min(self.tmax, t_supply))
```

C4. Kubernetes: Carbon Aware Scheduling with Taints/Tolerations + KEDA

Carbon intensity as a Prometheus metric (scraped hourly per region):

```
carbon_intensity_g_per_kwh{region="south"} 240
carbon_intensity_g_per_kwh{region="north"} 380
```

KEDA ScaledObject for a batch deployment

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: batch-trainer
spec:
  scaleTargetRef:
    name: batch-trainer
  pollingInterval: 60
  cooldownPeriod: 600
  minReplicaCount: 0
  maxReplicaCount: 100
  triggers:
  - type: prometheus
    metadata:
      serverAddress: http://prometheus:9090
      metricName: carbon_score
      threshold: "250"          # scale when CI < 250 g/kWh (inverted via query)
      query: |
        (300 - carbon_intensity_g_per_kwh{region="south"})
```

Affinity to lowcarbon nodes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: batch-trainer
spec:
  replicas: 0
  selector:
    matchLabels: {app: trainer}
  template:
    metadata:
      labels: {app: trainer}
    spec:
      nodeSelector:
        region: south
      tolerations:
      - key: carbon/high
        operator: Exists
        effect: NoSchedule  # avoid nodes tainted as highcarbon
```

C5. Grafana/Prometheus: PUE Panel Query Examples

Instant PUE

```
(sum by(dc) (power_total_kw{feed="utility"}))
  /
(sum by(dc) (power_it_kw))
```

Daily Average PUE

```
avg_over_time(
  (sum by(dc) (power_total_kw{feed="utility"}) / sum by(dc) (power_it_kw))[1d:5m]
)
```

C6. Python + CVXPy: Toy MPC for Cooling Setpoints

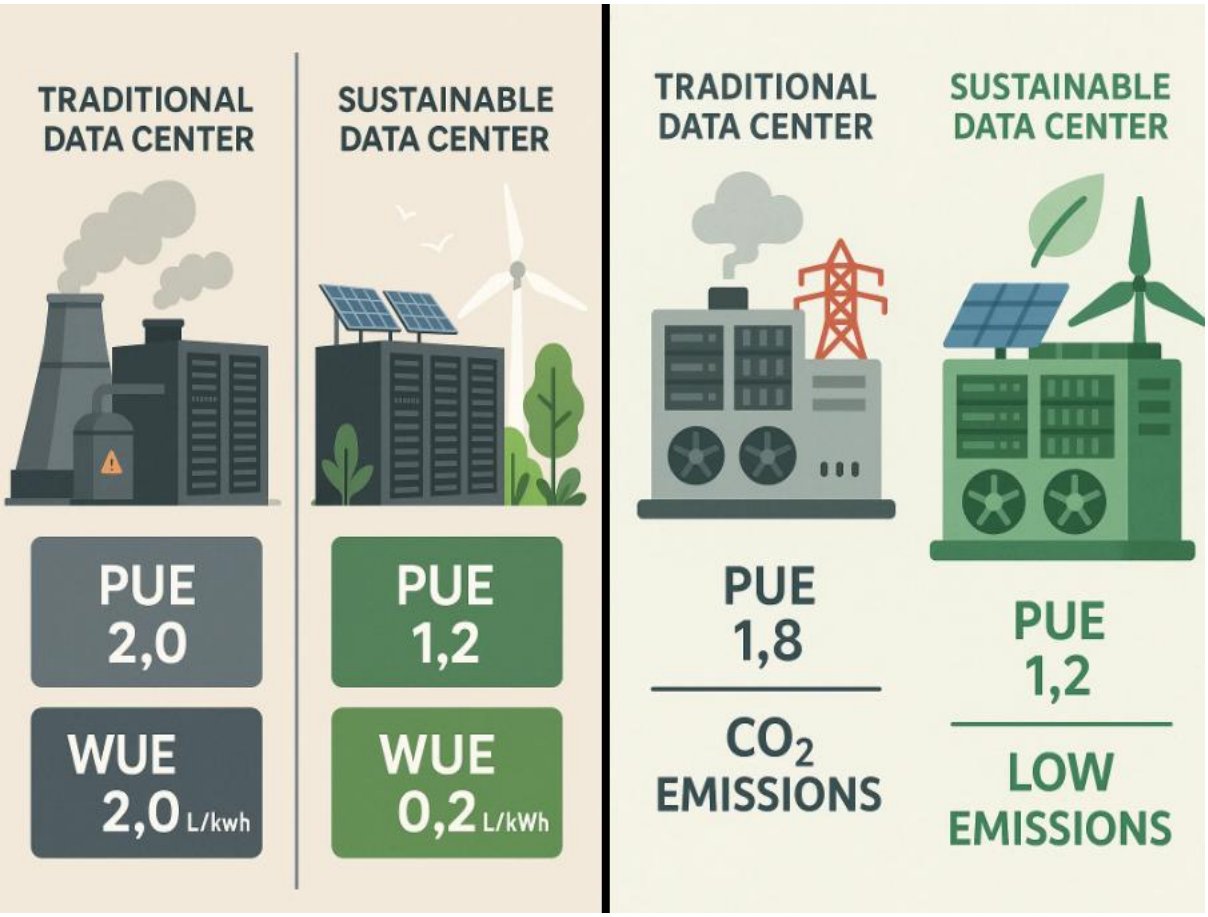
```
# pip install cvxpy numpy
import cvxpy as cp, numpy as np
# decision variables over horizon H
H = 8
T_supply = cp.Variable(H)
Fan = cp.Variable(H)
Pump = cp.Variable(H)

# parameters (example numbers)
IT_kw = np.array([450,470,500,520,510,490,480,460])
ambient = np.array([30,31,33,35,34,32,31,30])

# models (affine approximations)
facility_kw = 0.2*IT_kw + 0.05*Fan + 0.04*Pump + 0.3*(35 - T_supply)
water_l_kwh = 0.0 + 0.2*(33 - T_supply).clip(min=0)
carbon_g_kwh = 500 - 4*(ambient - 30)

energy = cp.sum(facility_kw)
water  = cp.sum(water_l_kwh)
carbon = cp.sum(cp.multiply(carbon_g_kwh, facility_kw/1000))

obj = cp.Minimize(1.0*energy + 0.02*water + 0.001*carbon)

constraints = [
   T_supply >= 20, T_supply <= 27,
   Fan >= 0, Fan <= 1.0,
   Pump >= 0, Pump <= 1.0
]
prob = cp.Problem(obj, constraints)
prob.solve(solver=cp.ECOS)
print('Optimal T_supply:', T_supply.value)
```

C7. Bash: Redfish Power Telemetry Poller (to InfluxDB line protocol)

```
#!/usr/bin/env bash
HOST=$1; TOKEN=$2
resp=$(curl -s -k -H "X-Auth-Token: $TOKEN" https://$HOST/redfish/v1/Chassis/1/Power)
it_kw=$(echo "$resp" | jq '.PowerControl[0].PowerConsumedWatts' | awk '{print $1/1000}')
now=$(date +%s)
echo "it_power,host=$HOST value=$it_kw $now" >> /var/lib/telegraf/it_power.lp
```

8. CONCLUSION

The exponential growth of digital services has made sustainable computing infrastructure a critical priority for industry, academia, and governments alike. Energy-efficient data centers are no longer optional; they are foundational to reducing operational costs, minimizing environmental impacts, and ensuring the long-term viability of global digital ecosystems.



This research has explored design principles and strategies for building sustainable data centers, including advanced cooling techniques, renewable energy integration, carbon-aware workload scheduling, and AI-driven optimization. By implementing holistic metrics such as PUE, WUE, and CUE, operators can gain a comprehensive understanding of energy and resource efficiency. Moreover, integrating predictive control systems, demand-response mechanisms, and hybrid renewable-storage architectures further enhances sustainability.

The findings underscore that sustainability in computing is not achieved through a single technology but rather through a synergistic approach that combines hardware innovation, software intelligence, and renewable energy utilization. Future data centers must be designed as adaptive, carbon-aware, and resilient infrastructures capable of self-optimizing in real-time.

Ultimately, sustainable computing infrastructure represents more than an engineering challenge—it is a societal imperative. As the demand for cloud services, artificial intelligence, and high-performance computing accelerates, the transition to energy-efficient, low-carbon, and environmentally conscious data centers will define the next generation of digital progress.

REFERENCES:

[1] N. V. A. Ravikumar, R. S. S. Nuvvula, P. P. Kumar, N. H. Haroon, U. D. Butkar and A. Siddiqui, "Integration of Electric Vehicles, Renewable Energy Sources, and IoT for Sustainable Transportation and Energy Management: A Comprehensive Review and Future Prospects," 2023 12th International Conference on Renewable Energy Research and Applications (ICRERA), Oshawa, ON, Canada, 2023, pp. 505-511, doi: 10.1109/ICRERA59003.2023.10269421

[2] .A. K. Bhaga, G. Sudhamsu, S. Sharma, I. S. Abdulrahman, R. Nittala and U. D. Butkar, "Internet Traffic Dynamics in Wireless Sensor Networks," 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023, pp. 1081-1087, doi: 10.1109/ICACITE57410.2023.10182866.

[3] Butkar, U., Pokale, N.B., Thosar, D.S. and Potdar, S.R. 2024. THE JOURNEY TO SUSTAINABLE DEVELOPMENT VIA SOLAR ENERGY: A RECAP. ShodhKosh: Journal of Visual and Performing Arts. 5, 2 (Feb. 2024), 505–512. DOI:https://doi.org/10.29121/shodhkosh.v5.i2.2024.2544.

[4] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in Proc. ACM SIGCOMM, 2009, pp. 123–134.

[5] Butkar, U. D., Sabale, R. P., & Patil, P. C. (2024). THE FUTURE OF TRANSPORTATION: A COMPREHENSIVE ANALYSIS OF ELECTRIC VEHICLES AND THEIR IMPACT ON SUSTAINABILITY, ECONOMY, AND SOCIETY. ShodhKosh: Journal of Visual and Performing Arts, 5(3), 900–907. https://doi.org/10.29121/shodhkosh.v5.i3.2024.3431

[6] The Green Grid, "PUE: A comprehensive examination of the metric," The Green Grid, White Paper, 2012.

[7] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 732–794, 2016.
[8]Uamakant, B., 2017. A Formation of Cloud Data Sharing With Integrity and User Revocation. International Journal Of Engineering And Computer Science, 6(5), p.12.
[9] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers," Journal of Parallel and Distributed Computing, vol. 71, no. 6, pp. 732–749, 2011.
[10] H. Xu, C. Feng, and C. Liu, "Decentralized proactive scheduling for data center demand response," in Proc. IEEE INFOCOM, 2017, pp. 1–9.
[11] Microsoft, "Project Natick: Cloud computing under the sea," Microsoft Research, 2020. [Online]. Available: https://natick.research.microsoft.com
[12] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in Proc. ACM SIGMETRICS, 2005, pp. 303–314.